3D INDOOR SCENE SEGMENTATION USING CONSENSUS CLUSTERING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

FURKAN MEVLÜT KÜÇÜKDEMIR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MODELLING AND SIMULATION

AUGUST 2020

Approval of the thesis:

3D INDOOR SCENE SEGMENTATION USING CONSENSUS CLUSTERING

Submitted by **FURKAN MEVLÜT KÜÇÜKDEMIR** in partial fulfillment of the requirements for the degree of **Master of Science in Modelling and Simulation Department, Middle East Technical University** by,

Date: 28.08.2020

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: FURKAN MEVLÜT KÜÇÜKDEMIR

Signature :

ABSTRACT

3D INDOOR SCENE SEGMENTATION USING CONSENSUS CLUSTERING

Küçükdemir, Furkan Mevlüt
M.S., Department of Modelling and Simulation
Supervisor: Assoc. Prof. Dr. Yusuf Sahillioğlu

August 2020, 56 pages

In this study, we propose an indoor scene segmentation method which utilizes consensus clustering. Unlike most of the recently offered methods in literature, our approach does not rely on deep learning techniques. Therefore, it does not require a large dataset for training and spending a lot of time to learn a valid model is not necessary. In the first step of our algorithm, we construct uniform and cotangent Laplace operators. Then, we compute differential coordinates using them and global point signatures using the eigenbasis of cotangent Laplace operator. In the next step, we use these coordinates and global point signatures as features and run k-medoids multiple times to create an ensemble. With the help of Partial Evidence Accumulation Clustering method, which is a consensus clustering approach, we obtain the final segmentation. Optionally, we offer an interactive segmentation mechanism to our users, in case any adjustment on the final segmentation is needed. The key idea of our approach is to use a specialized function to compute distance between feature points, which takes the scene geometry into account by using surface properties such as normals. At the end of the thesis, we also present both qualitative and quantitative evaluation of our method and show that it outperforms some of the existing techniques, quantitatively.

Keywords: indoor scene segmentation, cotangent Laplace operator, consensus clustering

ÖZ

ORTAK KÜMELEME KULLANARAK 3B İÇ MEKAN SAHNE BÖLÜTLENMESİ

Küçükdemir, Furkan Mevlüt
Yüksek Lisans, Modelleme ve Simülasyon Bölümü
Tez Yöneticisi: Doç. Dr. Yusuf Sahillioğlu

Ağustos 2020 , 56 sayfa

Bu çalışmada, iç mekan sahnelerinin bölütlenmesi sorununa ortak kümeleme kullanan bir yöntem öneriyoruz. Literatüre sunulan son çalışmaların çoğunluğunun aksine, önerdiğimiz yöntem derin öğrenme tekniklerine dayanmamaktadır. Bu nedenle, büyük veri kümelerine ihtiyaç duymamaktadır ve geçerli bir model öğrenmek için çok zaman harcamasına gerek yoktur. Algoritmanın ilk adımında, tekdüze ve kotanjant Laplace operatörleri oluşturuyoruz. Sonra, Laplace operatörlerini kullanarak diferansiyel koordinatları ve kotanjant Laplace operatörünün eigen bazını kullanarak evrensel nokta imzalarını hesaplıyoruz. Sonraki adımda, bu koordinatları ve evrensel nokta imzalarını nitelik olarak kullanıyor ve topluluk oluşturmak için çok kere k-medoids çalıştırıyoruz. Bir ortak kümeleme yaklaşımı olan Kısmı Kanıt Biriktirme Kümelenmesi yönteminin yardımıyla, nihai bölütlemeyi elde ediyoruz. İsteğe bağlı olarak, eğer nihai bölütleme üzerinde herhangi bir düzenleme gerekli görülürse, kullanıcılarımıza bir etkileşimli bölütleme mekanizması sunuyoruz. Yaklaşımımızın kilit fikri nitelik noktaları arasındaki mesafe hesaplanırken, normaller gibi yüzey özelliklerini kullanarak sahne geometrisini dikkate alan özelleşmiş bir fonksiyon kullanılmasıdır. Tezin sonunda, yöntemimizin hem nitel hem de nicel değerlendirmesini sunuyor ve varolan bazı tekniklerden nicel olarak daha üstün olduğunu gösteriyoruz.

Anahtar Kelimeler: iç mekan sahne bölütlemesi, kotanjant Laplace operatörü, ortak kümeleme

for all the trees of the valley, who once clapped their hands

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Assoc. Prof. Dr. Yusuf Sahillioğlu for his advice and guidance.

I would also like to thank my family for their continuous support.

This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under the project EEEAG-215E255.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
ARPACK	Arnoldi Package
DNN	Deep Neural Network
EAC	Evidence Accumulation Clustering
GCE	Global Consistency Error
GPS	Global Point Signature
HOG	Histogram of Oriented Gradients
LCE	Local Consistency Error
LIDAR	Laser Imaging Detection and Ranging
MRI	Magnetic Resonance Imaging
PAM	Partition Around Medoids
PEAC	Partial Evidence Accumulation Clustering
RGB-D	Red Green Blue-Depth
RI	Rand Index
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features

CHAPTER 1

INTRODUCTION

Segmentation of 3D models is one of the fundamental tasks in digital geometry processing and computer graphics. For 3D models which represent individual objects, this problem is well studied and many researchers in the field proposed solutions. See [4] for a general overview about these techniques.

1.1 Motivation and Problem Definition

In recent years, depth cameras, which are capable of scanning indoor and outdoor scenes, become more and more popular. Extracting meaningful information from these scanned scenes enable a lot of creative applications to emerge. For this reason, segmentation of these scanned environments, which may represent an indoor or an outdoor scene, gained much importance lately. Consider a domestic robot, for instance. It is crucial for the robot to recognize the environment and to interact with it [5]. Similarly, an autonomous vehicle must perceive pedestrians, traffic islands, traffic lights and so on [6]. Scene understanding may even be used to improve the life quality of visually impaired people [7] [8].

[9], [10] and [11] are some examples of the studies dealing with this problem which we discuss in Chapter 2. Aside from segmentation, these studies also provide semantic labeling which improves scene understanding substantially. What they share in common is the fact that they use deep learning. So, they require large datasets and hours of training to show their best performance. Most of the methods that are proposed in recent years utilize deep learning techniques in some way but there are still some research works to develop non-deep learning methods to solve the scene segmentation problem [12].

In the context of video games, an efficient solution to scene segmentation problem may help developers to incorporate new mechanisms to their 3D content creation pipeline. Such a solution, for example, would make it possible for players to scan their room with a depth camera and use their own goods in the virtual world of the video game. Moreover, combined with mixed reality methods, it may enable game developers to embed cutscenes which take place in players' surrounding real-world environment. Imagine your favorite video game character sitting on the couch next to you, for instance.

1.2 Proposed Methods and Models

In this paper we present a new approach for indoor scene segmentation problem which does not rely on deep learning methods. The key idea of our solution is to collect an ensemble of segmentations and generate the final segmentation which exhibits the consensus across the ensemble.

Briefly, our method consists of the following steps. In the first step, we compute the uniform Laplacian, cotangent Laplacian and the eigendecomposition of cotangent Laplacian. Then, we calculate differential coordinates based on uniform Laplacian and cotangent Laplacian (δ_{uni} and δ_{cot}, respectively) and global point signatures (GPS) for every face on the input scene. We use these two δ-coordinates and GPS as features and in the next step, we calculate the distance between every pair of faces in each feature space. After that, we run k-medoids clustering algorithm multiple times to construct the ensemble. In the last step, we run a consensus clustering algorithm to generate the final segmentation. After this step, we provide a recommendation mechanism in case any modification on the final segmentation is needed.

1.3 Contributions

The main contributions of our work are as follows:

- In this work, we propose a new method to partition a scene based solely on surface normals, which is called dihedral-based partitioning. It is mainly designed for noise-free, dense scenes. However, we show that it may also be useful on simplified and noisy ones. In our study, for instance, we use it as a heuristic to make a guess about the number of segments in a connected component of a scene.

- To the best of our knowledge, we propose the first method that utilizes consensus clustering for scene segmentation problem.

- Even if there are some interactive segmentation approaches proposed in the literature, we are first to offer a recommendation-based segmentation tool which recommends segments with different levels of granularity to the user.

1.4 The Outline of the Thesis

Outline of the remaining of this thesis is organized as follows:

Chapter 2 presents a summary of existing approaches in literature for mesh and scene segmentation tasks. Then, it briefly reviews interactive segmentation methods proposed so far. It continues with a brief explanation of local and global shape descriptors. In the last section, an overview of consensus clustering methods is given.

Chapter 3 explains the method proposed in this work in a detailed fashion. This chapter ends after addressing some remarks about the implementation.

Chapter 4 begins with a brief description of the dataset we have used for the experiments and compares it with some other datasets provided by various researchers. Then, it presents both qualitative and quantitative evaluation of our method. In the quantitative evaluation section, evaluation metrics proposed for segmentation algorithms are discussed. Finally, it gives a brief discussion about the results.

Chapter 5 summarizes preceding chapters and presents some of the limitations of the proposed method. It concludes after offering some ideas for future works.

CHAPTER 2

RELATED WORK AND BACKGROUND

2.1 Scene Segmentation

Meaningful decomposition of digital objects (e.g., humans, animals) is one of the fundamental problems in computer vision tasks such as object recognition. These objects are commonly discretized as mesh structures and there are mesh segmentation methods that handle their partitioning. The aim of mesh segmentation algorithms is to find a partitioning which divides the input mesh into meaningful segments. In the last two decades, a number of studies have been proposed for this problem in literature, some of which are fully-automatic [13] and some others are supervised [14]. Moreover, several studies, which discuss these methods and related topics in a detailed fashion, have been published [15] [16] [17].

In real world, collection of objects make up scenes. It is possible to digitize real-world scenes using scanners (also called depth sensors or depth cameras) which are able to sense depth information. As the scanning device technologies such as LIDAR, Asus Xtion and Microsoft Kinect [18] advance and new techniques emerge to improve the quality of scans, more applications from a variety of fields start to utilize 3D models for their purpose [19]. Scene understanding, which involves the analysis of an indoor/outdoor scene and gathering semantic information about the environment, is a fundamental task in computer vision which lies at the heart of many fields such as autonomous driving [6], augmented reality [20] and robotics [21]. In the last decade, a lot of progress has been made in scene understanding. However, we are still far from fully understanding a scene and it remains as one of the major problems in computer vision. Scene segmentation is one of the scene understanding problems and it concerns the partitioning of a given scene into meaningful and reasonable regions. Semantic scene segmentation is a related problem and besides finding the segments, it also involves labeling them accurately.

The way machines and humans perceive and interpret the visual information about their environment are quite different. From a machine's point of view, an image is just a large array of numbers or a scene is a collection of 3D points which store some properties such as location and color. In the eyes of a human, on the other hand, they contain more information and have a lot more meaning. In order to fill the gap between machines and people, researchers have proposed a variety of feature descriptors some of which are SIFT [22], HOG [23] and SURF [24].

Segmentation algorithms proposed to work in both 2D and 3D domains revolved

around hand-engineered shape descriptors until early 2010s. However, after the success of deep neural networks (DNNs) [25] in various fields including image classification [26] and speech recognition [27] become apparent, computer vision community also began to utilize them in segmentation tasks [28]. Nowadays, deep learning techniques have reached the state-of-the-art in semantic scene segmentation [29] and it is proven that they are able to handle problems in 3D domain, effectively [30]. However, they depend on large input datasets which should have a great variety of possible inputs in order to perform well [31]. Moreover, they are still expensive in terms of computational cost since large scenes contain a lot of data which should be given as input to DNNs. Because of that reason, some studies do not handle 3D data directly [30]. Instead, they reduce the resolution of input data or utilize cheaper representations.

There are various representations used to encode the scene data. Most popular ones of them can be listed as follows [32]:

- **Point Clouds:** As the name suggests, point clouds are collections of points in 3D space each of which has different attributes such as location and color. Scanning devices usually store raw scan data in this representation.
- **Voxels:** 3D scenes can be represented using cubic, volumetric grids. A unit sample of such grids are called voxels. Such discrete representation of the scene allows researchers to reduce the amount of data that should be processed. However, this may cause loss of information and detail.
- **Meshes:** Meshes are graphs which consist of vertices, faces and edges. In computer graphics, discrete 3D models usually represented as polygon meshes. Nevertheless, it may be necessary to use volumetric (e.g. tetrahedral) meshes for some tasks such as elastic body simulation [33].

There are studies which work on RGB-D images [34], as well. In these techniques, segmentation is accomplished in image space, where depth information is also available. The downside of them is that these techniques fail to understand and to utilize the geometry of the scene completely. Deep learning methods, which are used to solve various computer vision problems, require a regular input data format. Since point clouds and meshes contain variable amount of data units (i.e. points, faces, edges), it is a common approach to voxelize them [35] [36], even if some works in literature have proven that it is possible to use raw point clouds as input [10]. Another popular representation used in deep learning methods is multi-view images [37]. There are also some studies which utilize feature descriptors as their neural network input [38].

Scene segmentation methods can be categorized into two groups based on the environment: *indoor scene segmentation* [39] [40] and *outdoor (or urban) scene segmentation* [41] [42].

The study done by Koppula et al. [39] offers a solution which operates on 3D point clouds and the method they proposed works by over-segmenting the indoor scene and predicting labels for each segment. Their approach is also applied on a domestic robot to find objects in a room [43]. The main idea in Silberman et al.'s work [40] is that

objects scattered around an indoor scene (e.g. a room) are in relation with surrounding planar surfaces such as floor and walls. Their approach relies on finding those planar surfaces and interpreting their relation with nearby objects. They use RGB-D images as input and the first step in their solution is to find surface normals and planar surfaces. Then, they segment each plane using color and depth information. In the final step, they infer support relations and generate a hierarchical segmentation.

For outdoor segmentation, Martinovic et al. [41] proposed an efficient method which works completely on 3D data. More particularly, their work focuses on semantic facade parsing and it uses weak architectural principles (e.g. symmetry or alignment) to produce labelings. Another study on outdoor segmentation [42] uses both images and 3D point clouds to extract information about the environment. Their approach works by performing segmentation on multiple scales and then fusing information gathered from these multiple segmentations.

Compared to indoor places, outdoor environments are less complicated to work with since they are relatively flat-shaped and less complex. Indoor scenes, on the other hand, consist of a lot more objects and they are often irregularly shaped and arbitrarily placed. Indoor scans are likely to contain incomplete (i.e. partially scanned) regions due to occlusion.

2.2 Interactive Segmentation

Shape and scene processing methods can be classified as supervised, unsupervised and semi-supervised [44]. Supervised algorithms rely on training data and learn models to solve the problem at hand. In general, these methods produce most successful outputs. However, they depend on the quality and size of the training data, heavily. Unsupervised methods do not require any data to learn how to approach the problem. Nevertheless, they tend to rely on parameters, which are difficult to tune, in order to produce a generic model. Semi-supervised methods fall into between these two approaches.

In addition to supervised, semi-supervised and unsupervised segmentation algorithms, there are also some methods that depend on user assistance for segmentation process, which are called *interactive segmentation* techniques [45]. In the context of mesh segmentation, these techniques can be categorized into three classes [46]:

- **On-Boundary Brushes:** The first type of interactive mesh algorithms is the ones that use along-cut strokes which rely on the user specifying the cut boundaries [47]. These applications, however, require too much effort from the user.

- **In-Segments Brushes:** In this approach, the user is expected to draw sketches which roughly specifies the foreground and the background [48]. Then, cuts are applied on the mesh based on these sketches using various techniques including region growing algorithms [49] in real-time [50]. The downside of these approaches is that the user cannot fully control where the cuts will be applied [51].

- **Cross-Boundary Brushes:** This method works based on strokes provided by the user which give hints about both the location and the orientation of the boundaries [51]. Even though it offers more control to user over specifying boundaries, compared to in-segments brushes, it may require too many strokes to refine high quality cuts [46].

In literature, there are a few studies which utilize interactive tools for semantic scene segmentation. The method proposed in [9] utilizes voxel hashing [52], which enables it to work efficiently on a volumetric representation of the scanned scene. In this approach, the user wears a depth camera and as the user walks around, the environment is being reconstructed in real-time. Then, when the user interacts with the scene (i.e. by touching surrounding objects or by giving voice commands), the framework interprets user's actions to locate segments and to label them. As a result, users can obtain an annotated 3D model of their environment in real-time. Nguyen et al. [53] proposed an interactive tool which utilizes an automatic segmentation algorithm and passes segmentation results to the user for refinement and annotations of these segments. This tool allows users to use merge, split and extract operations for the refinement process. A recent study [54] offers a semantic labeling tool which is based on virtual reality. This work uses gamification to make it easier for people without any prior knowledge about semantic segmentation to interact with the environment in an entertaining way. Moreover, they propose a way to combine annotations produced by different users and to create an uncertainty map which can later be used to assess the quality of a segmentation.

2.3 Shape Descriptors

In computer vision and geometry processing, it is a common approach to represent shapes in a compact space and to use d-dimensional vectors, where d is generally an application dependent parameter. These d-dimensional vectors are called *shape descriptors* [2]. In literature, there are two types of shape descriptors: *local shape descriptors* (also known as *point signatures*) and *global shape descriptors*, which are also called *shape fingerprints*. Local shape descriptors are feature vectors defined for every node (i.e. vertex or face) on the mesh and they are mainly used in areas such as partial shape matching, segmentation, point correspondence estimation [44]. On the other hand, global shape descriptors are used as a representation of the whole shape. Global shape descriptors can be constructed by using local shape descriptors. Mainly, they appear in areas such as shape classification and shape retrieval.

In the early days of geometry processing, shape descriptors were mostly hand-engineered features and a frequently used kind of them is spectral shape descriptors [55]. Spectral shape descriptors (such as [56] [57]) are computed based on the eigenbasis of the Laplace-Beltrami operator, which is discussed in the following chapter. Wave Kernel Signature [57], Heat Kernel Signature [56] and their variants [58] [59] are examples of spectral shape descriptors and they are employed by many applications in geometry processing [60] [61] as well as in various fields such as graph processing [62], computational biology [63].

After the proliferation and success of deep learning approaches in various fields, many

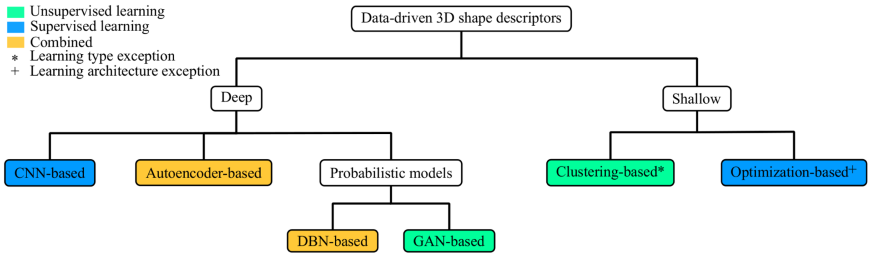

Figure 2.1: Algorithm Based Classification of 3D Shape Descriptors (Figure Source [2])

deep learning based shape descriptors are proposed [44]. The advantage of this approach is that it is possible to construct features from the input dataset automatically with no or little knowledge. The downside of them, however, is that they require large and information-rich input datasets to detect useful features. They are categorized in two groups as shown in Figure 2.1. They may also utilize hand-engineered shape descriptors and some works show that this kind of descriptors perform better compared to their hand-engineered counterparts [64].

Interested readers can find more detailed information in surveys published about the topic [65] [66] [67] [2].

2.4 Consensus Clustering

In literature, the collection of clustering results obtained by running one or more clustering algorithms (e.g. k-means [68]) several times is called ensemble. Developing unsupervised algorithms using a single clustering algorithm is difficult and solutions are prone to be unstable and weak based on the parameters used. The main rationale of ensemble clustering is to obtain better clustering results by combining the information gained by different clusterings of the input data [69]. This technique is also known as ensemble clustering and it is utilized in various areas of research and has proven to be successful [70] [71]. Compared to individual clustering algorithms, there are several advantages of using consensus clustering approach [72]:

- **Robustness:** The performance on different fields and data is better, on average.
- **Stability:** Ensemble methods are less sensitive to noise and outliers. Moreover, adverse effects of randomization used in individual clustering algorithms (e.g. random seed selections in k-means) are reduced.
- **Novelty:** Clusters which are unattainable by using a single clustering algorithm can be obtained.
- **Parallelization:** Since multiple clustering results are needed for ensemble methods, they are trivially parallelizable.

In the process of creating the ensemble, there are several approaches to improve the diversity among the clusterings [73] [74] [75]. The first option is to use different

algorithms for individual runs. Another option is to run those algorithms with different parameters. Running k-means with different values of k, for example, is a valid solution.

There are several types of algorithms that utilize consensus clustering approach and one of them is co-association matrix based algorithms. The co-association matrix can be defined as a pairwise similarity matrix whose entries, C_{ij}, denote the ratio of the number of clusterings that data points i and j placed in the same cluster to the total size of the ensemble [69]. Thus, its entries, which have a value between 0 and 1, show the similarity of a pair of data points. In literature, there are several solutions based on co-association matrices [76]. In this work, we use Partial Evidence Accumulation (PEAC) proposed in [77].

CHAPTER 3

METHOD

In this chapter, we describe core steps of our method. We start with our pre-processing step where we prepare the input scene for our method. We continue with the features we have used, which are differential coordinates (two of them; one is calculated with uniform Laplacian, δ_{uni}, and the other with cotangent Laplacian, δ_{cot}) and global point signatures (GPS). Then, we explain how we create the ensemble by applying k-medoids clustering multiple times. Later, we give the details of the consensus clustering algorithm which we use to extract a single segmentation out of the ensemble. Finally, we finish with the optional step, which is interactive segmentation.

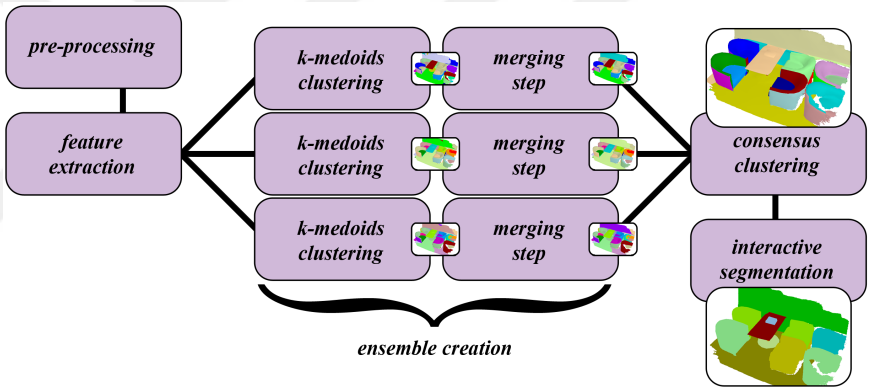

Figure 3.1: General overview of the proposed approach. We start by pre-processing step where we simplify the input scene. Then, we extract features (i.e. differential coordinates and global point signatures). After that, we create the ensemble by running k-medoids multiple times and send these clusters to consensus clustering to obtain the final segmentation. Optionally, we offer an interactive segmentation mechanism, in case the user wants to make some modifications to the final segmentation.

3.1 Pre-Processing

In general, scenes generated by depth camera scans are quite dense (See Figure 3.2). More explicitly, they may contain about a million faces or even more. However, it is not efficient, if possible, to make matrix calculations with that large raw scenes since it would require enormous amount of memory. In addition, these raw scenes may contain some parts which are not scanned clearly and those noisy parts may affect the

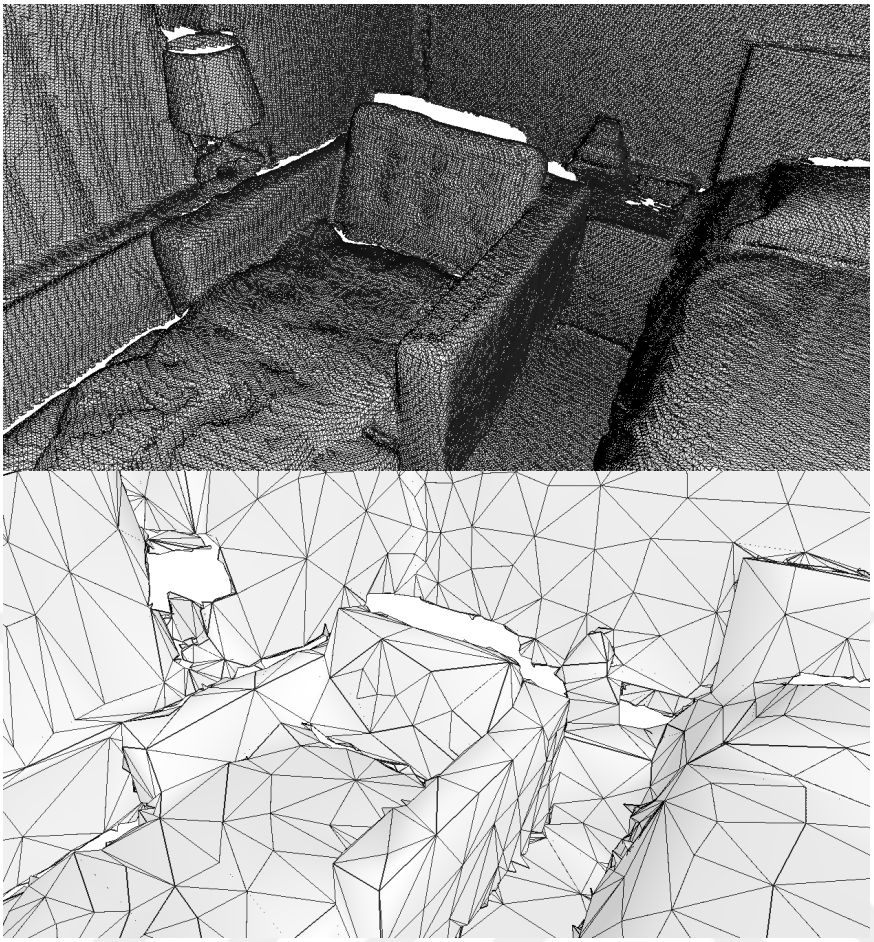

Figure 3.2: Raw scene which consists of 728536 vertices and 1468466 triangles (top) and simplified scene with 7368 vertices and 9816 triangles (bottom)

quality of segmentation since it does not represent a meaningful piece. In order to avoid such consequences, following steps are applied:

- First, identify connected components of the scene and sort them in descending order according to their face count. Then, mark components as valid until either at least %98 of the raw scene is covered or remaining components become smaller than %0.5 of the raw scene. In other words, eliminate components which contain too little number of faces in this step.
- Then, use Garland's QSlim [78] to simplify valid components of the scene and reduce the total number of faces to 10000.
- Lastly, discard connected components that are smaller than %1 of the simplified scene.

Note that the final segmentation is mapped back to the original scene at the end of the segmentation process and points which are eliminated in the pre-processing stage remain unlabeled.

After these pre-processing steps, the scene becomes ready for the rest of the method. However, it may consist of multiple connected components. Since we use parameters such as k (i.e. the number of clusters in k-medoids) and the desired number of final segments for the whole scene, we have to distribute these parameters to each connected component. The second parameter, the desired number of final segments, is used at consensus clustering step and it is the approximate number of segments to be present in the final segmentation expected by the user. There are several approaches, including:

- **Using the exact same parameters for all connected components** is the simplest way to handle the problem. However, it is not a desirable solution, since the size of components are very likely to be vary.
- **Scaling parameters according to the size of components** is a better approach. If we use the number of faces to determine the size of components (in terms of face or vertex count), parameters are distributed in a meaningful way for the majority of scenes.
- **Using a heuristic to detect the ideal distribution among the components** is the most appealing option but coming up with such a heuristic is challenging.

In our approach, we have used a heuristic which we call *dihedral-based partitioning*. The key idea behind dihedral-based partitioning is that if adjacent faces make roughly the same angle with one another in a certain part of the mesh, we assume that there is a meaningful partition. So, if we can find such partitions in each component and scale parameters according to the number of dihedral-based partitions found, we can make a reasonable distribution. Note that, aside from the distribution of parameters, dihedral-based partitions are used as recommendation segments in the interactive segmentation.

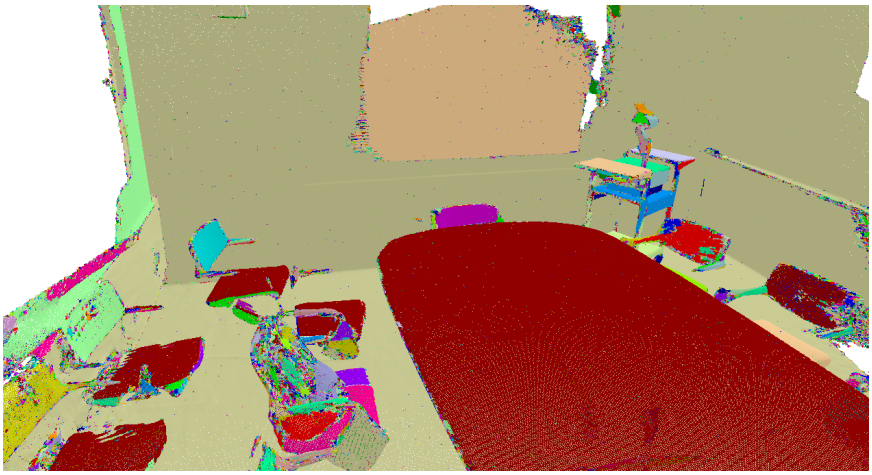

Figure 3.3: Partitions obtained using dihedral-based partitioning. Notice that $\frac{PartitionCount}{FaceCount}$ ratio is low for the table (i.e. large, red plane), which leads to a more reasonable distribution of k.

See Figure 3.3, for example. If we use the number of faces to distribute the number of final segments, the proposed method will be inclined to use a higher value for k to

divide the table, whereas we detect that it is a planar surface which represents a single partition. Moreover, if we prefer the former strategy for distribution, the method becomes sensitive to the resolution of the mesh.

Dihedral-based partitioning works as follows. Let s and t be indices of two adjacent faces and N_s and N_t their surface normals, respectively. We know that $-1 \leq \langle \mathbf{N}_s, \mathbf{N}_t \rangle \leq 1$, where $\langle ., . \rangle$ denotes the dot product. We divide the interval $[-1, 1]$ into D equal subintervals and get D slots. (In this thesis, we have used $D = 64$.) Thus, the length of each interval is

$$S = \frac{(1 - (-1))}{D} = \frac{2}{D} \tag{3.1}$$

In order to detect partitions with consistent angles, we check dihedral angle between each face and its 1-ring neighborhood. Then, each dihedral angle is assigned to a slot. If one of the slots has more assignments compared to all others, we choose the subinterval represented by that slot. If we cannot detect such a subinterval in 1-ring neighborhood, we search it in 2-ring neighborhood and so on. After detecting the subinterval, we traverse adjacent faces and mark them as part of the same dihedral-based partition, recursively. Algorithm 1 lists all steps in detail.

3.2 Laplace Operator

Laplace operator (also called *Laplacian*, briefly) is frequently used in various applications in geometry processing, machnine learning and other areas [79] [80] [81] [82] [83].

As a geometry processing tool, Laplacians are very useful for several reasons. For example, they reduce the dimensionality of the problem to the number of eigenfunctions [61]. More importantly, the Laplacian eigenbasis has nice features which allows us to build an orthonormal basis that reflects the geometry of the mesh [84] [85]. Eigenvalues of the Laplace-Beltrami operator can be thought as frequencies and eigenvectors of it can be considered as the value of functions evaluated at vertices on the mesh. As the corresponding eigenvalues of eigenvectors become smaller, they transform into smoother and more slowly varying functions.

Over the years, many discretizations are proposed for Laplacian including uniform Laplacian [86] and cotangent weighting scheme [87] [88] [89] [90] [91]. The uniform Laplacian (also called combinatorial Laplacian) depends only on the connectivity of the graph (i.e. mesh) and does not encode any other information (e.g. the spatial distribution of vertices) about the mesh. Because of that reason, it may be different for two different triangulations of the same exact mesh. Therefore, it is not the best choice in many applications, even if it is easy to compute and to understand. Still, it is used in some areas such as mesh compression [92]. On the other hand, cotangent weighting scheme is numerically consistent and more accurate since they preserve several properties of the geometry such as edge lengths and angles [90]. In our work, we use the discretization based on cotangent weighting scheme proposed by [93].

Algorithm 1 Dihedral-Based Partitioning

Input: Triangulated Scene Mesh M; Number of Slots, D
Output: Partitions Per Face P; Number of Partitions; C

1- Initialize$P$
2- C <- 0
3- $slotInterval$ <- $\frac{2}{D}$
4- Initialize slots
5- foreach face f in M do
6- if f is not labeled then
7- Initialize a queue, Q
8- Push f into Q
9- repeat
10- $angleFound$ <- $false$
11- repeat
12- q <- Pop next face from Q
13- foreach face a in $Adjacent(q)$ do
14- Calculate $\langle \mathbf{N}_q, \mathbf{N}_a \rangle$ and put it in corresponding slot
15- Push a into Q_{next}
16- end
17- until Q is empty
18- if maximum number of angles in a slot, U, is unique then
19- $limits_{min}$ <- U_{min} - $\frac{slotInterval}{2}$
20- $limits_{max}$ <- U_{max} + $\frac{slotInterval}{2}$
21- $angleFound$ <- $true$
22- end
23- Q <- Q_next
24- until $angleFound$ is $true$
25- P_f <- C
26- Initialize a queue, T
27- Push f into T
28- repeat
29- t <- Pop next face from T
30- foreach face a in $Adjacent(t)$ do
31- if $\langle \mathbf{N}_t, \mathbf{N}_a \rangle$ falls between $limits_{min}$ and $limits_{max}$ then
32- P_a <- C
33- Push a into T
34- end
35- end
36- until T is empty
37- Increment C
38- end
39- end
40- return P and C

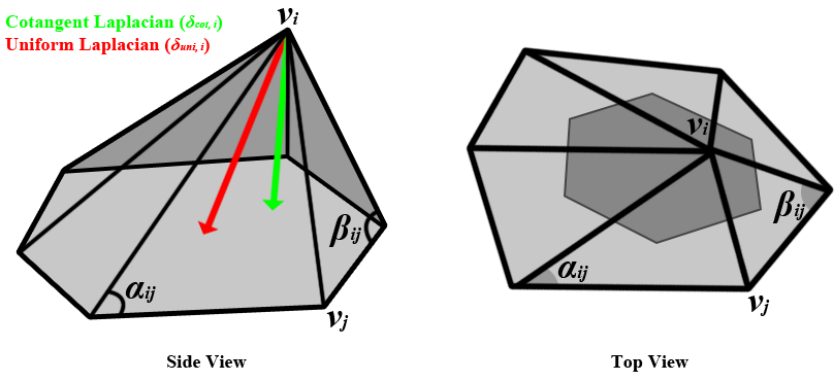

Figure 3.4: δ-coordinates based on uniform and cotangent Laplacians. Notice that cotangent Laplacian takes angles into account whereas uniform Laplacian points to the center of 1-ring neighbours.

Uniform Laplacian, L^{uni}, is defined as [85]:

$$L_{ij}^{uni} = \begin{cases} d_i & \text{, if i = j} \\ -1 & \text{, if i and j are adjacent} \\ 0 & \text{, otherwise} \end{cases}$$

, where d_i is the valence (i.e. number of 1-ring neighbors) of vertex i.

According to cotangent weighting scheme, cotangent Laplacian, L^{cot}, is defined as follows:

$$L_{ij}^{cot} = \begin{cases} \sum_k \frac{m_{ik}}{s_i} & \text{, if i = j} \\ \frac{-m_{ij}}{s_i} & \text{, if i and j are adjacent} \\ 0 & \text{, otherwise} \end{cases}$$

, where s_i denotes the area of Voronoi region (i.e. the shaded area in Figure 3.4, top view – see [94] for details) and

$$m_{ij} = \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2}$$

, when vertices i and j are adjacent. Otherwise, $m_{ij} = 0$.

L^{cot} can be factorized into two matrices as $L^{\text{cot}} = A^{-1} \times W$, where W is a sparse symmetric matrix and A is positive-definite diagonal matrix and their elements are (see Figure 3.4):

$$w_{ij} = \begin{cases} \frac{\cot \alpha_{ij} + \cot \beta_{ij}}{2} & \text{, if i and j are adjacent} \\ 0 & \text{, otherwise} \end{cases}$$

$$a_{ij} = \begin{cases} s_i & \text{, if i = j} \\ 0 & \text{, otherwise} \end{cases}$$

We can calculate δ-coordinates $\delta_{\mathrm{uni},i}$ and $\delta_{\mathrm{cot},i}$ for a vertex v_i as follows:

$$\delta_{\mathrm{uni},i} = L^{\mathrm{uni}} \times v_i \tag{3.2}$$

$$\delta_{\mathrm{cot},i} = L^{\mathrm{cot}} \times v_i \tag{3.3}$$

It is possible to find eigenbasis of L^{cot}, by solving the following generalized eigenvalue problem:

$$W\phi_i = \lambda_i A\phi_i \tag{3.4}$$

More detailed information about theoretical aspects of the Laplace-Beltrami Operator can be found in [95] [96] [97].

3.3 Global Point Signature

In this section, we describe *Global Point Signature (GPS)* [98] which is the shape descriptor used in our method. Let M be an m-dimensional compact manifold and p a point on M. In order to compute GPS(p), we have to find Laplace-Beltrami eigenbasis, i.e. eigenfunctions and eigenvalues of the Laplace-Beltrami, as explained in the previous section. Let ϕ_i and λ_i be i^{th} eigenfunction and its corresponding eigenvalue, respectively and assume eigenvalues to be sorted such that

$$0 = \lambda_0 < \lambda_1 < \lambda_2 < \cdots < \lambda_i < \cdots \tag{3.5}$$

Then, we define GPS(p) as follows:

$$\mathrm{GPS}(p) = \left(\frac{1}{\sqrt{\lambda_1}}\phi_1(p), \frac{1}{\sqrt{\lambda_2}}\phi_2(p), \frac{1}{\sqrt{\lambda_3}}\phi_3(p), \cdots \right) \tag{3.6}$$

In plain words, GPS(p) is an infinite-dimensional vector whose components are values of eigenfunctions evaluated at p divided by the square root of their corresponding eigenvalues. Notice that the 0^{th} eigenfunction and eigenvalue are not used in the computation since they do not give any useful information. Note that λ_0 is 0 and $\phi_0(p)$ is a constant-valued vector for all M [85].

Since Laplace-Beltrami operator is invariant under isometric deformations of the shape and GPS is based on Laplace-Beltrami eigenbasis, GPS holds the same property. Nevertheless, the signs of eigenfuntions are not reliable (i.e. they may switch) and that may not be desirable for certain applications. The ordering of eigenfunctions also suffers from the same problem [99].

As stated in Equation 3.6, Global Point Signature (GPS) is infinite-dimensional. However, in practice, GPS is truncated, i.e. its only first k components are used. In this thesis, we have used $k = 50$.

3.4 Creating the Ensemble

A collection of indices gathered by running a clustering algorithm on a dataset n times is called *ensemble*. One of the most used clustering algorithms for this purpose is k-*means* [68]. Euclidean distance between two n-dimensional data points is defined as follows:

$$\text{euclidean}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.7}$$

The main aim of k-means is to minimize the mean Euclidean distance between data points and the centroid of their clusters. In theory, it does not guarantee to give accurate results and it is quite sensitive to outliers. However, in practice, it is used in variety of applications because it is easy to understand and to implement and it runs fast. When creating the ensemble, the chosen clustering algorithm is run multiple times. As a result, it is crucial to choose a clustering algorithm which is not expensive in terms of computational time. For this reason, we tried to use an improved version of k-means called k-*means*++ [100] which utilizes a strategy when choosing the initial centroids (i.e. seeds) instead of choosing them arbitrarily.

Nevertheless, k-means++ is unsuitable for our purposes for two reasons:

- We use geodesic distance which requires us to make use of adjacency between data points. Because of that, when we utilize δ_{cot} as feature, for example, we require centroids to be a subset of V_{cot}, where V_{cot} is the set of vertices in cotangent Laplacian based differential coordinates. Otherwise, the clustering algorithm may end up in an infinite loop.
- We use a weighted Euclidean distance in feature (i.e. δ-coordinates and global point signatures) space to compute distance between adjacent data points. The distance function defined for δ_{cot}, for example, can be defined as follows:

  $$\text{distance}(x, y) = \text{euclidean}(\delta_{\text{cot},x}, \delta_{\text{cot},y}) * \text{cost}(x, y) \tag{3.8}$$

  , where x and y are indices of two adjacent faces and

  $$\text{cost}(x, y) = (A + B * \langle \mathbf{N}_x, \mathbf{N}_y \rangle)^C, \tag{3.9}$$

  where A, B and C are coefficients, N_x and N_y are surface normals. We have tested our solution with various values for the coefficients (i.e. A, B, and C). By replacing the δ_{cot} term in Equation 3.8, we get distance function for other features.

Because of the reasons listed above, we have selected k-medoids instead of k-means++. The main idea behind k-medoids is similar to k-means. However, unlike k-means, centroids have to be one of the data points in k-medoids and they are called *medoids*. This property makes k-medoids less sensitive to outliers in data points.

There are several algorithms proposed for k-medoids clustering in the literature. Most popular of them is called Partition Around Medoids (PAM) [101]. Nevertheless, it suffers from high computational complexity which makes it unsuitable to run on large datasets. Thus, we have preferred to use the algorithm proposed by Park and Jun [102]. In [102], they list four different approaches to choose initial medoids. In our work, we have tested all four approaches and noticed that the optimal strategy is to choose initial medoids completely in random (i.e. Method 1 in [102], Section 3.4).

Algorithm 2 shows the full procedure.

Algorithm 2 K-Medoids Clustering [102]

Input: Data Points, P
Output: Medoid Index Per Data Point, M

1- Compute pairwise distance matrix, D, where D_{ij} represent
   distance between P_i and P_j
2- Select initial medoids among P, randomly
3- $sumOfDistances_{prev}$ <- ∞
4- $converged$ <- $false$
5- repeat
6- Assign every data point to its closest medoid and update M
7- Calculate sum of distances between every data point and its medoid
8- if $sumOfDistances$ equals to $sumOfDistances_{prev}$ then
9- $converged$ <- $true$
10- else
11- Update medoids by finding the data points whose sum of distance
   to other data points in the cluster is minimum
12- end
13- $sumOfDistances_{prev}$ <- $sumOfDistances$
14- until $converged$ is $true$
15- return M

The first step of the algorithm is to build a pairwise distance matrix D, where D_{ij} represent the distance from data point i to data point j. The distance values are calculated as stated in Equation 3.8. In order to compute distance between non-adjacent faces, we have used Dijkstra's shortest path algorithm [103]. Note that the matrix D is calculated once in our method. Since we use same distance matrix for all runs of k-medoids, we do not compute it over and over again. The remainder of steps are followed in all runs, separately.

After building the distance matrix, we select initial medoids randomly among the data points and begin to iterate. In each iteration, we assign each data point to the closest medoid and compute the sum of distance between each data point and its medoid. If the sum of distance does not change, we stop iterating and return the cluster result. Otherwise, we update medoids for every cluster by finding the data point whose sum

of distance to all other data points in the cluster is minimum. Then, we proceed to next iteration and so on.

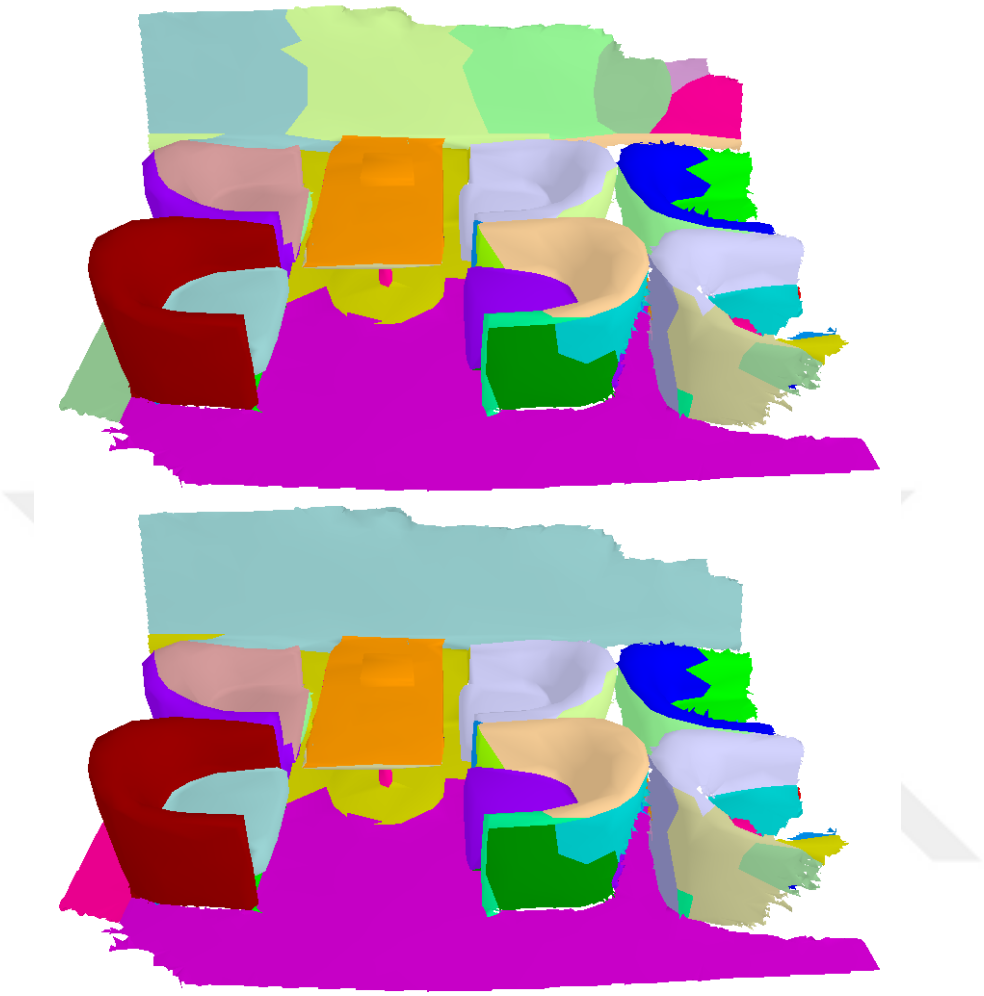

Figure 3.5: The clusters returned by k-medoids before (top) and after (bottom) the merging step. The number of clusters on the wall is reduced from 7 to 1, without loss of any useful information.

Before adding the cluster results returned by k-medoids to the ensemble, we have one more step called merging. Results after k-medoids may oversegment a cluster due to the parameter k. For example, see Figure 3.5, where k-medoids returns 7 clusters for the wall even though it is possible to merge these 7 clusters and consider the whole region as a single meaningful part. So, in this step, we check if there exist such clusters, namely neighboring segments with almost the same normal vectors, and merge them if there are any.

For this purpose, we build scatter matrix of each segment separately and find their eigenvectors and eigenvalues. If we denote these eigenvalues in decreasing order with λ_0, λ_1 and λ_2, we calculate $flatness = \frac{\lambda_2}{\lambda_0}$ for each segment as explained in [1]. Then, we examine the pairwise relation of each segment with its adjacent segments. If the following criteria are satisfied, we merge the segments in question:

- If the flatness of both segments are below the predefined threshold (0.008 in our work). Note that flatness value becomes lower as the segment gets more flat.

- If the dot product of eigenvectors corresponding to smallest eigenvalues are above the threshold (0.99 in our work). These eigenvectors represent the direction with lowest variance (i.e. surface normal, in case the segment is planar) and we want them to point at roughly the same direction.

- Finally, we check surface normals at the seam (i.e. all edges between two segments). If at least half of the face normals point at similar directions and mean dot product of those faces around the seam is above the threshold (0.99 in our work), we merge the segments in question.

3.5 Consensus Clustering Using Partial Evidence Accumulation

Let F be the number of faces of the input scene, E be the ensemble size, k_i be the number of clusters in i^{th} run of k-medoids and c_{fi} be the index of f^{th} face in i^{th} run of k-medoids, where $i \leq E$ and $f \leq F$. After constructing the ensemble, we get a key, which consists of E integers, for every face. Formally,

$$\text{key}(f) = (c_{f1}, c_{f2}, c_{f3}, \cdots, c_{fE}) \tag{3.10}$$

In order to find base segments, we group faces by their key such that two distinct faces s and t are in the same base segment if and only if the equality $key(s) = key(t)$ holds. In other words, all faces in a base segment have the exact same key. We will use B to denote the number of base segments and b_{ij} refers to the cluster index of i^{th} base segment in the j^{th} run of k-medoids.

We use base segments since they enable us to use smaller matrices, which reduces the amount of memory used and makes computations faster in the consensus clustering step. Considering that the expected number of base segments is much less than the number of faces, this approach provides a significant gain in terms of performance.

In the consensus clustering step, we aim to find a segmentation which combines the information from the whole ensemble. If we define a distance measure which gives the distance between two segmentation as:

$$d^2(x, y) = \sum_{a,b \in \{1,2,3,\cdots,B\}} \hat{w}_{ab}(\mathbb{I}_{xa=xb} - \mathbb{I}_{ya=yb})^2, \tag{3.11}$$

where

$$\mathbb{I}_p = \begin{cases} 1, & \text{if } p \text{ is true} \\ 0, & \text{otherwise} \end{cases} \tag{3.12}$$

and $\hat{w}_{ab}$ is the sum of the pairwise weights of all faces in the base segments a and b. Note that all pairwise weights are non-negative.

Then, we can write the optimization problem as follows:

$$y^* \in \underset{y \in \{1,2,3,\cdots,M\}^B}{\operatorname{argmin}} \sum_{i=0}^{E} d^2(b_{*i}, y), \tag{3.13}$$

where M is the desired number of clusters in the segmentation.

This optimization problem can be written as follows (for details, see [104]):

$$Y^* \in \underset{Y \in S}{\operatorname{argmin}} \|C - Y^T Y\|_W^2, \tag{3.14}$$

where $\|.\|_W$ stands for the *weighted Frobenius norm*, W is an $B \times B$ weight matrix such that $W_{ab} = \hat{w}_{ab}$, C is an $B \times B$ matrix whose entries represent the co-occurrence of base segments. In other words, c_{ij} is the number of times that base segment i and base segment j fall in the same cluster divided by the ensemble size. So, $0 \leq c_{ij} \leq 1$.

In order to solve Equation 3.14, we fed the consensus clustering algorithm with the co-association matrix C and the weight matrix W, where w_{ij} equals to pairwise product of the surface area of base segments i and j. Thus, we emphasize the importance of accurate segmentation of large base segments.

Consensus clustering algorithm we have used in our work, which is based on *Evidence Accumulation Clustering (EAC)* paradigm. Detailed information about the algorithm can be found in [104], [77] and [105]. Note that it returns an $M \times B$ matrix Y where M is the desired number of segments, B is the number of base segments and $0 \leq Y_{ij} \leq 1$. In [104], the index of the row containing the maximum value is assigned to the index of corresponding base segment in final segmentation. We, however, use a threshold and if the maximum value for a base segment is less than 0.5, we do not merge that base segment with another one and it remains as itself in the final segmentation. Therefore, the number of clusters in the final segmentation may be greater than M.

The segments found by the consensus clustering algorithm does not have to be connected (i.e. it may merge two base segment in the final segmentation even if they are not connected). In the final step of our method, we separate such segments, if there are any.

3.6 Interactive Segmentation

The final segmentation obtained with our unsupervised method may not be perfectly suitable for the user's need. At this point, we provide an interactive segmentation mechanism for users in case they want to make changes on the final segmentation. For this purpose, we offer three types of recommendation segments and two operations.

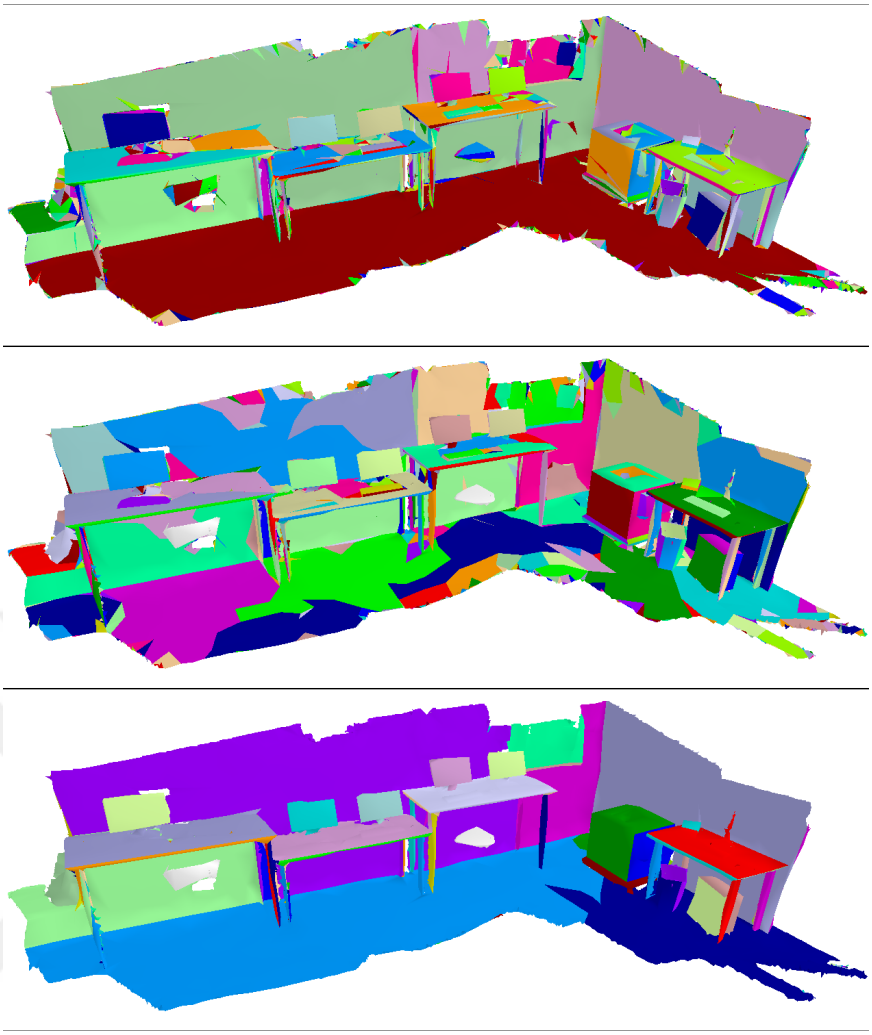

Figure 3.6: Recommendation segments used in the interactive segmentation. From top to bottom: Partitions obtained using dihedral-based partitioning, base segments, final segments.

Those recommendation segments, which can be previewed by the user before usage (See Figure 3.6), are:

- Partitions found by the dihedral-based partitioning step. These segments represent the most granular recommendations. In other words, they are suitable for modifying the fine details on the scene.
- Base segments, which are defined in previous sections. Most of the time, granularity of base segments is adequate and the user can edit almost the whole scene using them only. They rarely require split operations on them.
- Segments in the final segmentation. These are useful to change the segment of large parts. For example, if an armchair is divided into four parts (i.e. cushion, backrest and two arms) but the user wants to represent the whole armchair with a single segment, this type of recommendation segments are the most suitable

one.

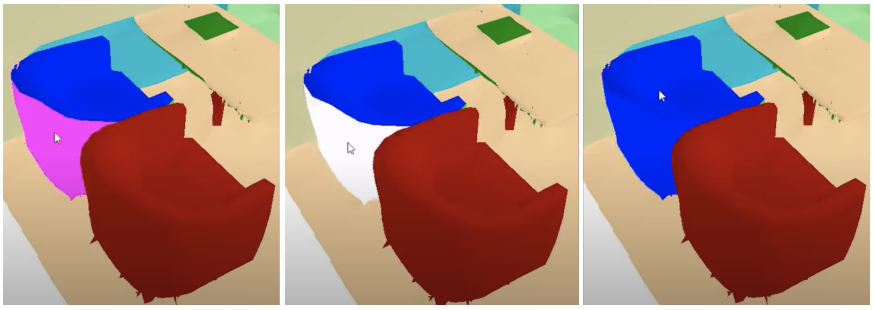

Figure 3.7: User interaction through *merge* operation to update the segmentation of the chair. When the user hovers over a segment, it starts to blink (top, notice the position of the cursor). Clicking the left mouse button, dragging the mouse over another segment and releasing the left mouse button merges two segments in question (bottom).

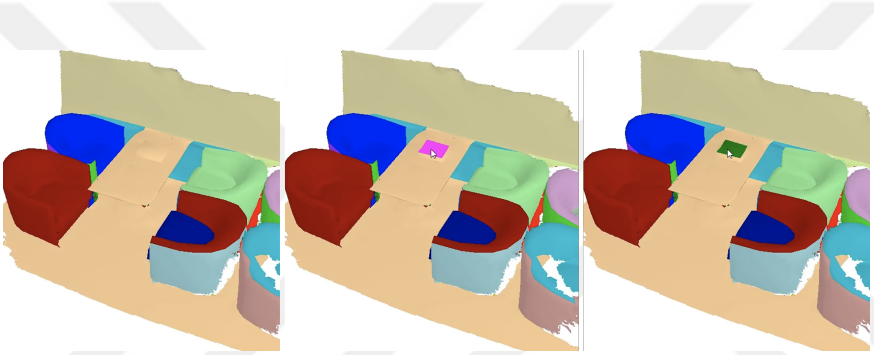

Figure 3.8: Demonstration of *split* operation. When the user hovers over a segment, it starts to blink (top, notice the position of the cursor). Clicking the right mouse button splits the highlighted segment and it becomes a new, separate segment (bottom).

When the user hovers the mouse over the scene, the selected type of recommendation segment at the tip of the cursor start to blink (Figure Figure 3.7 and 3.8). Using the highlighted part, we allow users to make two kinds of operations:

- **Merge**: If the user clicks the left mouse button, drags the mouse and release it, the highlighted part is merged into the segment of the face which is pointed by the cursor when the left mouse button is released.
- **Split**: If the user clicks the right mouse button when there is a highlighted part, that highlighted part is split from its segment and becomes a new segment on its own.

In addition to them, users can undo/redo if they want to take modifications back. Note that segments formed in interactive segmentation do not have to be connected.

After the user is done with the modifications, the segmentation can be saved for future use. Interactive segmentation is easy to use. It takes around 15 minutes for first time

users to complete the segmentation of an indoor scene (e.g. a living room) and it reduces around 5 minutes in later uses. However, we did not investigate the usability of it extensively and further studies are needed.

An example full session of interactive segmentation can be watched at our accompanying video linked here:
https://drive.google.com/file/d/1IBHdrZKsgpz38sYdm-HkCWriI6RoZhi4/view.

3.7 Implementation Details

In this section, we list some details about the implementation of the method.

- We have implemented our solution in C/C++.
- For matrix manipulations, we have used Eigen software package [106].
- When finding the Global Point Signature embedding of the scene, we have used ARPACK's [107] *dsaupd* and *dseupd* routines to compute eigenfunctions and eigenvalues of Laplace-Beltrami operator. Before ARPACK, we used Spectra [108] which is a sparse matrix library based on Eigen. However, its implementation for the generalized eigenvalue problem is not completely suitable for our purposes. Because of that reason, it runs drastically slow compared to ARPACK.
- In the consensus clustering step, we have used Rota Bulò's implementation of the optimization algorithm [104] without any modification.

CHAPTER 4

RESULTS AND EVALUATION

In this chapter, we present the experiments we conducted to assess our work. We begin by introducing the dataset we have used in the experiments. Then, we share our qualitative evaluation where we examine the effects of using different values for several parameters. After that, we show the results of quantitative evaluation where we compare our method with an existing approach based on several metrics. Finally, we conclude by discussing the outcomes of our experiments.

4.1 Dataset

Figure 4.1: A sample of scenes from the SceneNN dataset (Figure Source: [3])

While searching a dataset to use in our experiments, we took two important features into account:

- **Scene Format:** Some datasets contain scenes as raw RGB-D images [109] [40] [110], or as point clouds [39] [111] [112]. Since our method works on triangulated meshes, using such datasets would require us to run a surface reconstruction algorithm. However, we do not want to add another step between data and our method, since we want our work to be easy-to-compare with other scene segmentation methods. Applying a surface reconstruction algorithm would bring another layer, which is irrelevant to the segmentation method used.

- **Annotation:** We want to compare our method with existing approaches both qualitatively and quantitatively. For quantitative evaluation, however, we need to have ground-truth labels on the scenes in the dataset.

Table 4.1: Comparison of Several Datasets

Dataset	Scene Format	Annotation	Real-World Scan
[39]	Point Cloud	✓	✓
[109]	RGB-D Image	✓	✓
[40]	RGB-D Image	✓	✓
[113]	RGB-D Image	X	✓
[111]	Point Cloud	Partial	✓
[1]	Triangulated Mesh	X	✓
[112]	Point Cloud	✓	✓
[114]	Triangulated Mesh	X	✓
[110]	RGB-D Image	✓	✓
[115]	Triangulated Mesh	✓	X
[3]	Triangulated Mesh	✓	✓

As shown in Table 4.1, we considered several datasets and only two of them ([115] and [3]) satisfied our requirements. Even though [115] offers properly annotated, triangulated meshes, it consists of synthetic data. For our experiments, we preferred SceneNN [3], which is created from real-world indoor scenes. More RGB-D datasets and details about them can be found in [116].

SceneNN is a dataset which contains 100 scenes. All of them are fully annotated and scenes include offices, kitchens, living rooms, bedrooms and more. Each scene contains from 7 to 63 objects and each object has two kinds of annotations: *instance-level* and *class-level*. In our experiments, we use instance-level annotations as the ground-truth segmentation. Some examples from the dataset are shown in Figure 4.1.

In order to use these scenes with our method, we have to apply pre-processing steps as explained in Chapter 3. Raw scenes contain over a million of faces and we simplify them to reduce the face count to ten thousand, at most. As a result, a significant amount of detail is lost (See Figure 4.2). Moreover, if a scene contains regions that are not fully scanned, those regions become extremely noisy (See Figure 4.3). Since we want our method to be fully automatic, we do not fix these issues.

4.2 Qualitative Evaluation

In this section, we present a sample from results of our method. We evaluate our results in terms of segments' ability to represent a meaningful real-world objects. For

Figure 4.2: Comparison of raw (top) and simplified (bottom) scenes in terms of detail

this purpose, we selected and used 18 scenes from the dataset. We show their segmentations for several different cases. Our main purpose is to show how the outcome of our method changes based on the features used and to this end, we run it with three distinct features: *global point signatures (GPS)* and *differential coordinates* calculated with *uniform Laplacian (δ_{uni})* and *cotangent Laplacian (δ_{cot})*. We have also tested our method with two different cost functions which are defined as follows:

$$f_1(x, y) = (1.0 - 0.0 * \langle \mathbf{N}_x, \mathbf{N}_y \rangle)^{1.0} \tag{4.1}$$

$$f_2(x, y) = (1.0 - 1.0 * \langle \mathbf{N}_x, \mathbf{N}_y \rangle)^{1.50} \tag{4.2}$$

Results are shown in Figure 4.4. Note that for all runs, we set the desired number of final segments to 50, minimum value of k to 100, maximum value of k to 125 (a random value in this interval is chosen for each run of k-medoids), and dimension of the GPS domain to 50. Besides, segmentations presented in the left part of Figure 4.4 do not utilize interactive segmentation since this step is optional.

On the other side, we used the method proposed in [1] to produce segmentations for

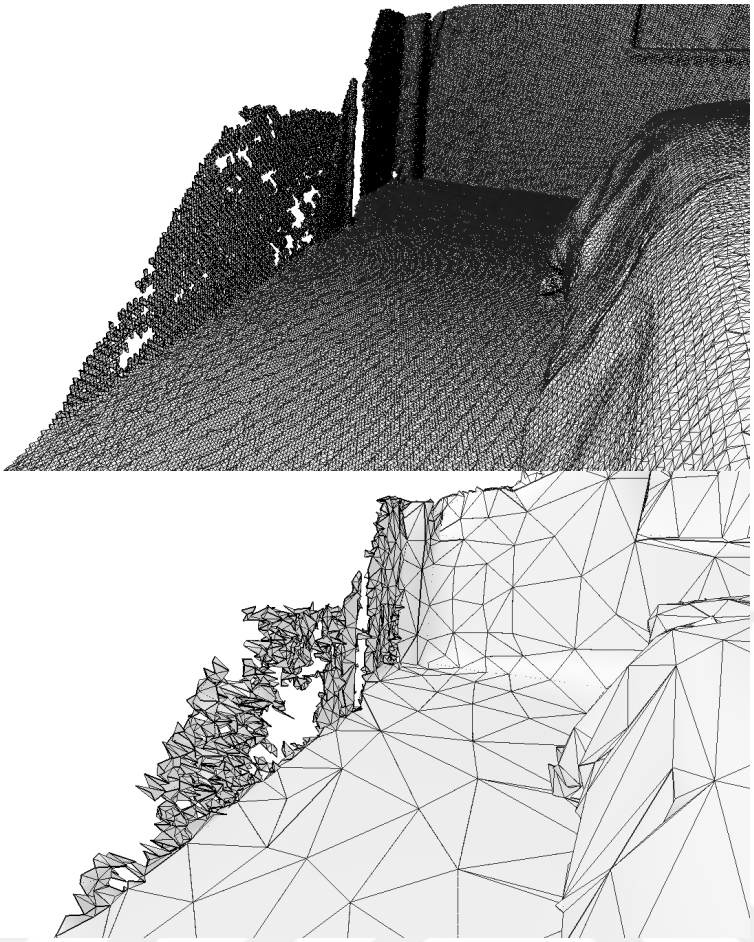

Figure 4.3: Partially scanned regions before (top) and after (bottom) the simplification

both scenes. In [1], however, the main focus is not to find a segmentation which labels every region of the scenes. Instead, they try to discover certain objects and label them accordingly. In order to make it more comparable with our method, we disabled their *hard thresholding* step (See [1] for more details). In addition to that, for comparison, we use the over-segmented scenes, which are put in a bag to rank and to detect certain objects in later steps, in the original algorithm. The over-segmented scenes are obtained 10 times by using different threshold values. Thus, we use 11 segmentations produced by [1] in our evaluation. [1]'s final segmentations are listed in Figures 4.5 and 4.6. Additionally, a sample of over-segmented scenes is shown in Figure 4.7.

4.3 Quantitative Evaluation

Stating the objective quality of a segmentation algorithm is difficult, since it may vary from one observer to another depending on their point of view. In addition to that, a segmentation of a certain model may be too fine or too coarse based on the application in which the model is used. Because of these reasons, there were no

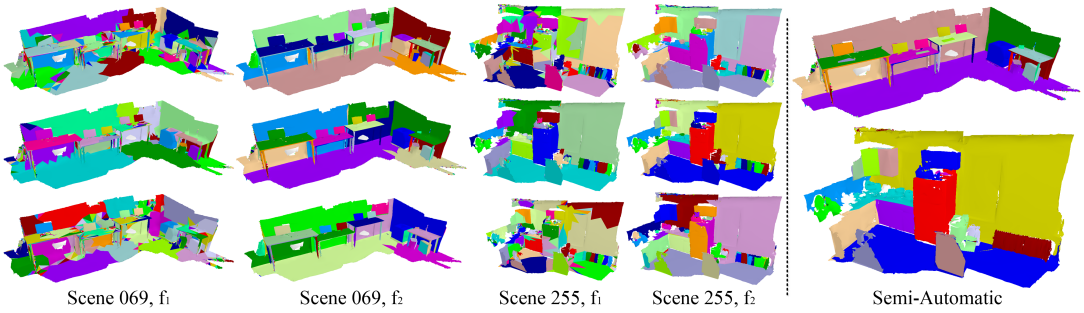

Figure 4.4: (Left) Final segmentations (as the outcome of the fully-automatic part of our method) obtained with our approach using f_1 and f_2. Each row utilizes a different feature as follows: top: δ-coordinates based on uniform Laplacian (δ_{uni}), middle: δ-coordinates based on cotangent Laplacian (δ_{cot}), bottom: global point signatures. (Right) Our segmentations of these two scenes (Scene 069 and 255) after user interaction.

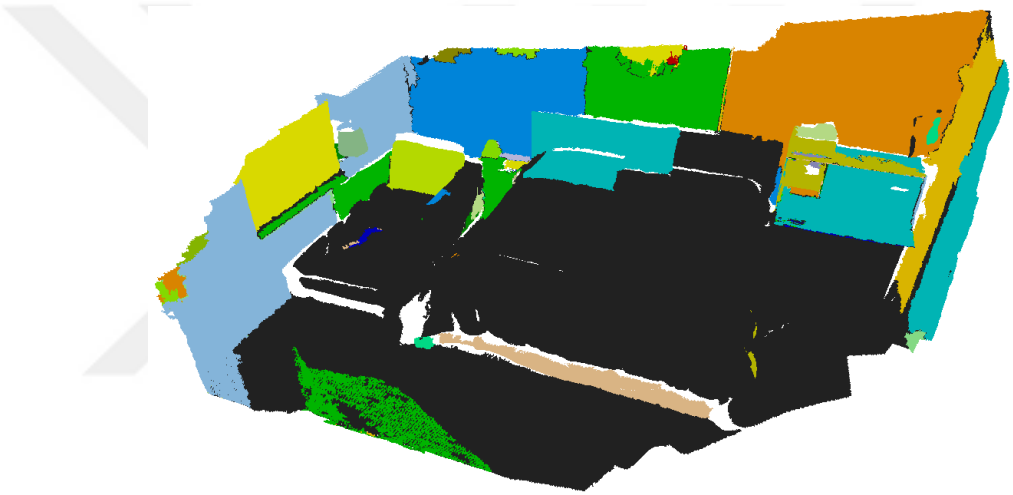

Figure 4.5: Final segmentation result for Scene 016 using [1]'s method. See our results in Figure 4.10 for comparison.

generalized, standard way to evaluate segmentation algorithms until 2009. Instead, researchers proposed some evaluation techniques based on the context. [117], for example, handles the problem in the domain of Magnetic Resonance Imaging (MRI), whereas [118] offers a way to make assessment based on the texture stretch for texture mapping applications. In 2009, two studies ([119] and [120]), which deal with this problem, are published, concurrently. Their approach is quite similar and [119] offers Global Consistency Error (GCE) and Local Consistency Error (LCE) as evaluation metrics. In addition to these metrics, [120] discusses Cut Discrepancy, Hamming Distance and Rand Index (RI).

Let S_1 and S_2 be two distinct segmentations of a mesh M.

- The *Cut Discrepancy* [121] metric measures the distance between the cuts in the first segmentation and the cuts in the second segmentation. Intuitively, it

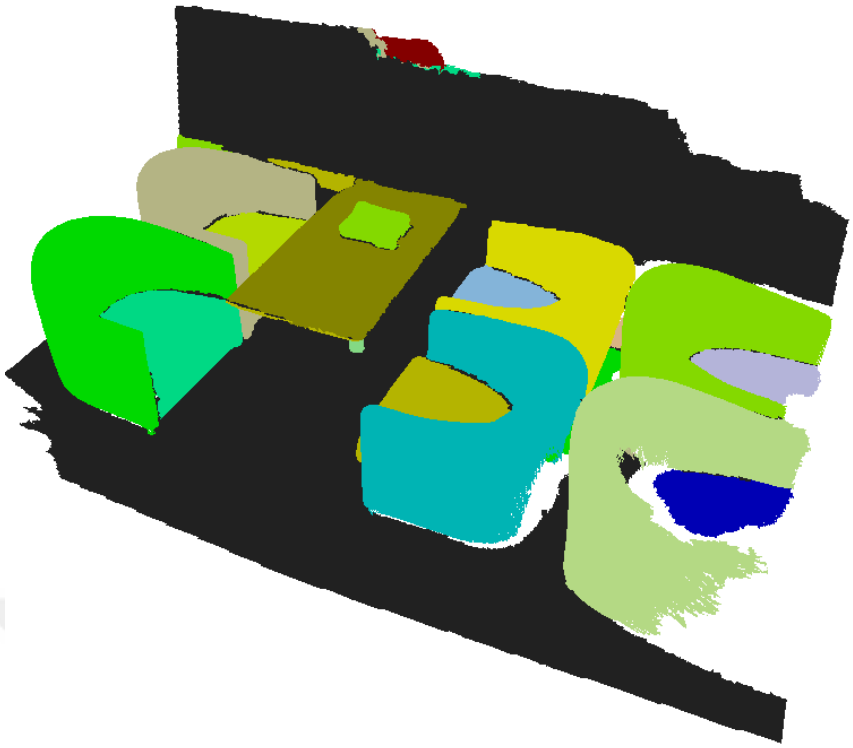

Figure 4.6: Final segmentation result for Scene 066 using [1]'s method. See our results in Figure 4.11 for comparison.

shows how well boundaries in each segmentation align. However, it is sensitive to segmentation granularity.

- *Directional Hamming Distance*, $\mathrm{D_H}(S_1, S_2)$, can be defined as the sum of the set differences between each segment in S_1 and the best corresponding segment in S_2. Then, *Hamming Distance* [121] can be defined as:

$$\mathrm{HD}(S_1, S_2) = \frac{\mathrm{D_H}(S_1, S_2) + \mathrm{D_H}(S_2, S_1)}{2 * A}, \tag{4.3}$$

  where A is the total surface area of the mesh. The downside of Hamming Distance is the same with Cut Discrepancy. In other words, it is also sensitive to segmentation granularity.

- The *local refinement error* of a vertex v_i between S_1 and S_2 is defined as follows:

$$\mathrm{L_{3D}}(S_1, S_2, v_i) = \frac{\|\mathrm{R}(S_1, v_i) \backslash \mathrm{R}(S_2, v_i)\|}{\|\mathrm{R}(S_1, v_i)\|}, \tag{4.4}$$

  where $\mathrm{R}(S, v)$ stands for the subset of vertices that share the same segment with vertex v in segmentation S, $\backslash$ for set differencing and $\|x\|$ the number of elements in the set x. Intuitively, this measure gives the ratio of the number of

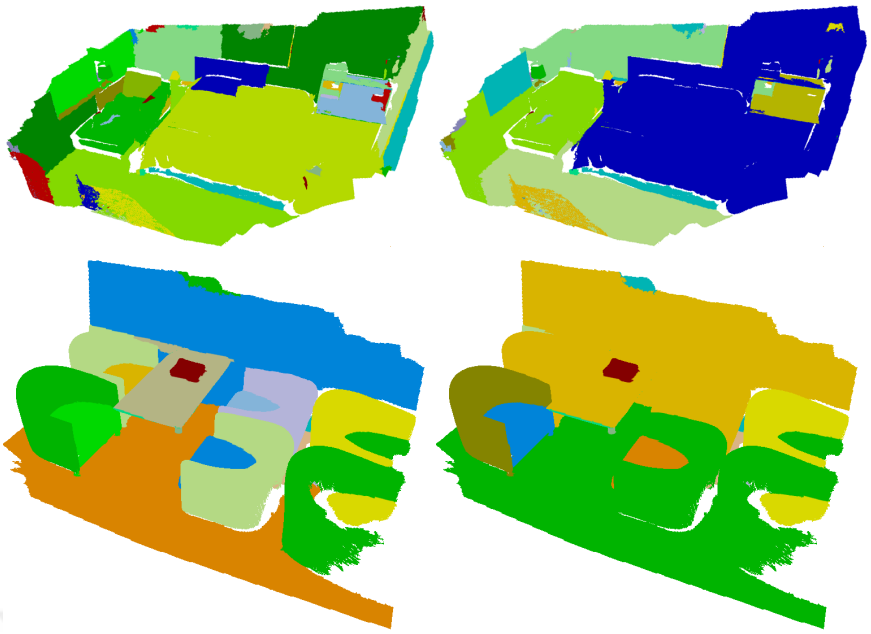

Figure 4.7: Oversegmentation of Scene 016 (top) and Scene 066 (bottom) for smallest (left) and largest (right) threshold values as used in [1]'s method.

non-shared vertices between two segmentations. In this definition, vertices are used. However, it is also possible and valid to use faces instead of vertices.

Using the local refinement error, the *Global Consistency Error* and *Local Consistency Error* [122], which account for hierarchical differences in segmentations, can be defined in the following way:

$$\mathrm{GCE}(S_1, S_2) = \frac{1}{N} min\{\sum_i \mathrm{L_{3D}}(S_1, S_2, v_i), \sum_i \mathrm{L_{3D}}(S_2, S_1, v_i)\}, \tag{4.5}$$

$$\mathrm{LCE}(S_1, S_2) = \frac{1}{N} \sum_i min\{\mathrm{L_{3D}}(S_1, S_2, v_i), \mathrm{L_{3D}}(S_2, S_1, v_i)\}, \tag{4.6}$$

where N is the number of vertices.

- The last metric, *Rand Index* [123], measures the probability of two faces being in the same segment or in different segments in both segmentations. Let s_i^1 and s_i^2 denote the label of face i in segmentation S_1 and S_2, respectively. Then,

$$\mathrm{RI}(S_1, S_2) = \binom{2}{N}^{-1} \sum_{i,j;i<j} \{[C_{ij}P_{ij} + (1 - C_{ij})(1 - P_{ij})]\}, \tag{4.7}$$

where C_{ij} evaluates to 1 iff s_i^1 equals to s_{ij}^1 and P_{ij} evaluates to 1 iff s_i^2 equals to s_{ij}^2. In literature, $1 - \mathrm{RI}(S_1, S_2)$ is used to keep metrics consistent. The most important advantage of the Rand Index is that it tolerates refinement only on certain parts of the mesh where humans tend to find ambiguous [124].

In short, Cut Discrepancy focuses on the errors on boundaries while others focus on the dissimilarities between regions. All of them provide similar relative performance [120]. In this work, GCE, LCE and the Rand Index are used for evaluation since Cut Discrepancy and Hamming Distance metrics are sensitive to segmentation granularity and tolerance to refinement is crucial for the scene segmentation problem.

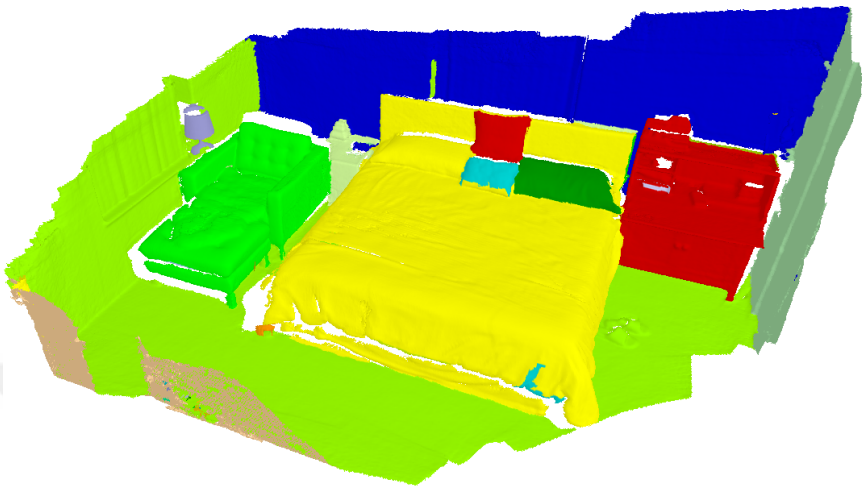

Figure 4.8: Ground-truth segmentation for Scene 016

After introducing evaluation metrics, we can assess the quality of our segmentations. For this purpose, we used 18 scenes from the dataset and before presenting quantitative results, we show a sample from their ground-truth segmentations (See Figure 4.8 and 4.9). As shown in figures, ground-truth segmentations are not fine detailed. For example, in 066, armchairs are considered as a single object instead of a combination of multiple segments (i.e. cushion, arms and backrest). For comparison, we also have semi-automatic (i.e. obtained with our interactive method) segmentations for both scenes. They are shown in Figure 4.10 and 4.11. Semi-automatic segmentations are done on simplified versions of scenes and they are slightly more fine grained.

Figure 4.10 and 4.11 show segmentations obtained with our method by using f_2 as the cost function and differential coordinates based on cotangent Laplacian as feature. Table 4.2 lists the average values of rand index, global consistency error and local consistency error for 18 scenes we have selected. Values for each scene are presented in Table 4.3, 4.4, 4.5 and 4.6, separately. Note that lower scores refer to better segmentation quality. Also note that values of "Semi-Automatic" are calculated using only two scenes. We calculated two scores for [1]'s results. One of them is computed based on the final outcome of their algorithm and the other one is the score of their over-segmented scenes (see the last paragraph of the previous section).

Note that we map the segmentations obtained with our method back to the original scene for quantitative analysis, since ground-truth segmentations are based on the original, raw scenes. Parts that are discarded in the simplification and pre-processing stages are left unlabeled and that does not cause any problems. Since we ran [1]'s method on the original scenes directly, it does not have such issues.

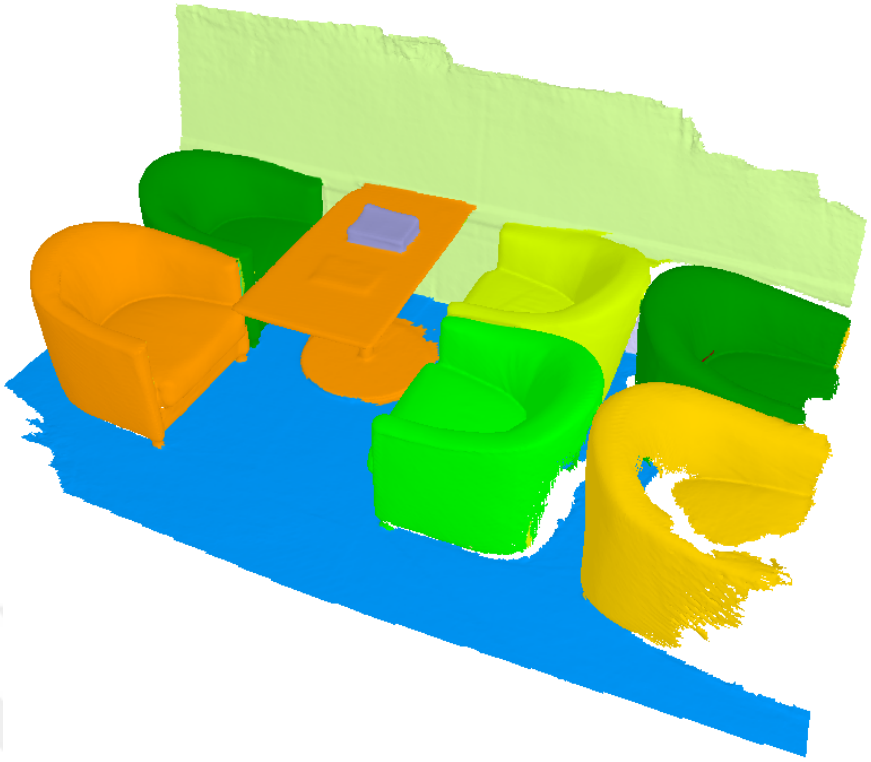

Figure 4.9: Ground-truth segmentation for Scene 066

4.4 Discussion

Evaluating the quality of a segmentation is not a trivial task. Even it is done by humans, there is no single correct way to divide a scene into segments. The level of detail, for example, may change the outcome drastically based on the way the subject (i.e. human or the algorithm) perceives the scene in question. When we compare our semi-automatic segmentations with the ground-truth ones, we observe a significant difference because of this reason.

When we observe quantitative results, we notice that the assessment done based on RI, GCE, LCE does not always reflect the quality of a segmentation. For example, we are able to produce segmentations whose GCE and LCE scores are lower than semi-automatic (i.e. obtained with interactive method) segmentations. However, semi-automatic segmentations are definitely better than these segmentations when compared visually (see Figure 4.12). We find RI to be the most reliable metric out of the metrics we have discussed so far and we use it for our comparisons.

When we examined the qualitative and quantitative results, we made the following observations:

- According to both qualitative and quantitative results, f_2 outperforms f_1. Therefore, calculating the distance between points in feature space is insufficient to produce a meaningful decomposition and cost functions help the algorithm to understand the scene geometry better. Moreover, we observe that our method

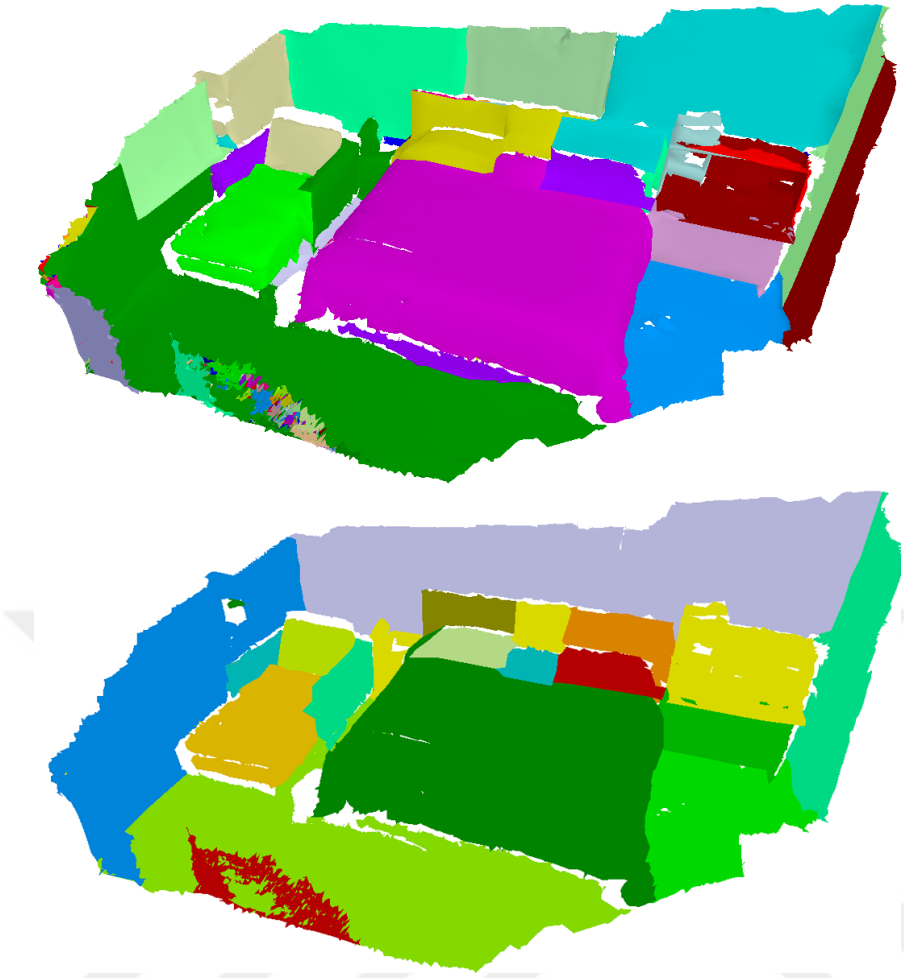

Figure 4.10: Top: Our fully-automatic segmentation results for Scene 016 with f_2 and δ_{cot}. Bottom: Our semi-automatic segmentation after user interaction.

becomes apt to apply more cuts and to produce more segments as the exponent term in cost functions lowers. This behaviour stems from the fact that as the exponent increases, the cost function becomes more tolerant to difference between surface normals of adjacent nodes. For example, if the dihedral angle between two faces is 45 degrees, then we scale the calculated distance by 0.16 and 0.12 when the exponent is 1.5 and 1.75, respectively.

- When we compare features, we notice that both GPS and differential coordinates based on cotangent Laplacian outperform differential coordinates based on uniform Laplacian. The reason behind this phenomenon is the fact that the cotangent discretization of Laplace operator encodes the geometry of the scene better. Since the uniform Laplacian uses only the connectivity of the graph (i.e. scene) and it depends heavily on its triangulation, its ability to represent geometric information is quite limited.

- When we compare segmentations obtained with f_1, we find that cotangent

Table 4.2: Summary of Evaluation Metrics. Note that values outside the parentheses are average for 18 scenes and values inside the parentheses give standard deviations

Segmentation	RI	GCE	LCE
Semi-Automatic	0.0417 (0.0117)	0.1458 (0.0242)	0.0951 (0.0206)
Ours $(f_2, \delta_{\cot})$	0.1124 (0.0312)	0.4081 (0.1136)	0.2918 (0.1063)
[1] (Over-segmented)	0.1413 (0.1015)	0.2658 (0.0803)	0.1262 (0.0463)
[1] (Final)	0.2382 (0.0914)	0.3556 (0.0663)	0.1315 (0.0401)

Table 4.3: Summary of Evaluation Metrics for Semi-Automatic

Scene ID	RI	GCE	LCE
016	0.0534	0.1699	0.1157
066	0.0300	0.1216	0.0745

Laplacian outperforms other features significantly. Even if the segments produced by cotangent Laplacian are not perfect, they are apparently more reasonable. This shows that the cotangent Laplacian is more powerful when it comes to catch geometric information. Nevertheless, in some scenes, GPS detects some details which are left unnoticed when Laplacians are used. Still, we think that the differential coordinates based on cotangent Laplacian is a better feature since it is only a 3-dimensional vector which makes it more efficient computation-wise compared to GPS, which usually uses more than 3 scalars.

- We find that the strength of GPS increases as the dimension of GPS domain increases. However, that is applicable only up to a threshold. After that, no significant improvements can be observed. In our experiments, we set it to 50.
- The desired number of segments, S, can be chosen according to desired level of granularity. In our experiments we set it to 50. For k_{min} and k_{max}, we realized that setting them to $2 * S$ and $2.5 * S$, respectively, gives pleasant results.
- Lastly, our approach outperforms the method proposed in [1], quantitatively (see Table 4.2).

Since our method depends on the surface normals a lot, it may fail when there is too much noise or the scene contains an object with a bumpy geometry. See Figure 4.13, for example. Our method divided the radiator shown in the lower right corner of Figure 4.13 into 9 segments because of this bumpy and noisy (due to simplification) surface.

Our final remark is about the tiny segments present in our segmentation results. The

Table 4.4: Summary of Evaluation Metrics for Ours $(f_2, \delta_{\text{cot}})$

Scene ID	RI	GCE	LCE
016	0.0914	0.2860	0.2388
057	0.1396	0.4324	0.2657
062	0.1030	0.3542	0.2536
066	0.1207	0.5286	0.4662
069	0.1380	0.5034	0.3897
071	0.1831	0.5161	0.3484
086	0.1166	0.5524	0.4329
109	0.1399	0.4135	0.3534
207	0.1019	0.5090	0.3995
223	0.1340	0.5425	0.3371
227	0.0530	0.2197	0.1350
234	0.1346	0.2626	0.1729
243	0.0601	0.3400	0.2101
249	0.0682	0.2755	0.1228
255	0.0922	0.4215	0.2473
273	0.1156	0.4654	0.3793
286	0.1202	0.5085	0.3745
522	0.1105	0.2152	0.1257

threshold added at the end of consensus clustering causes these tiny segments to show up. However, we do not prefer to lower the threshold, since that makes consensus clustering to merge semantically separate segments. Compared to unnecessary merges, tiny segments are seen less frequent. In addition, fixing the tiny segments issue is trivial using interactive segmentation.

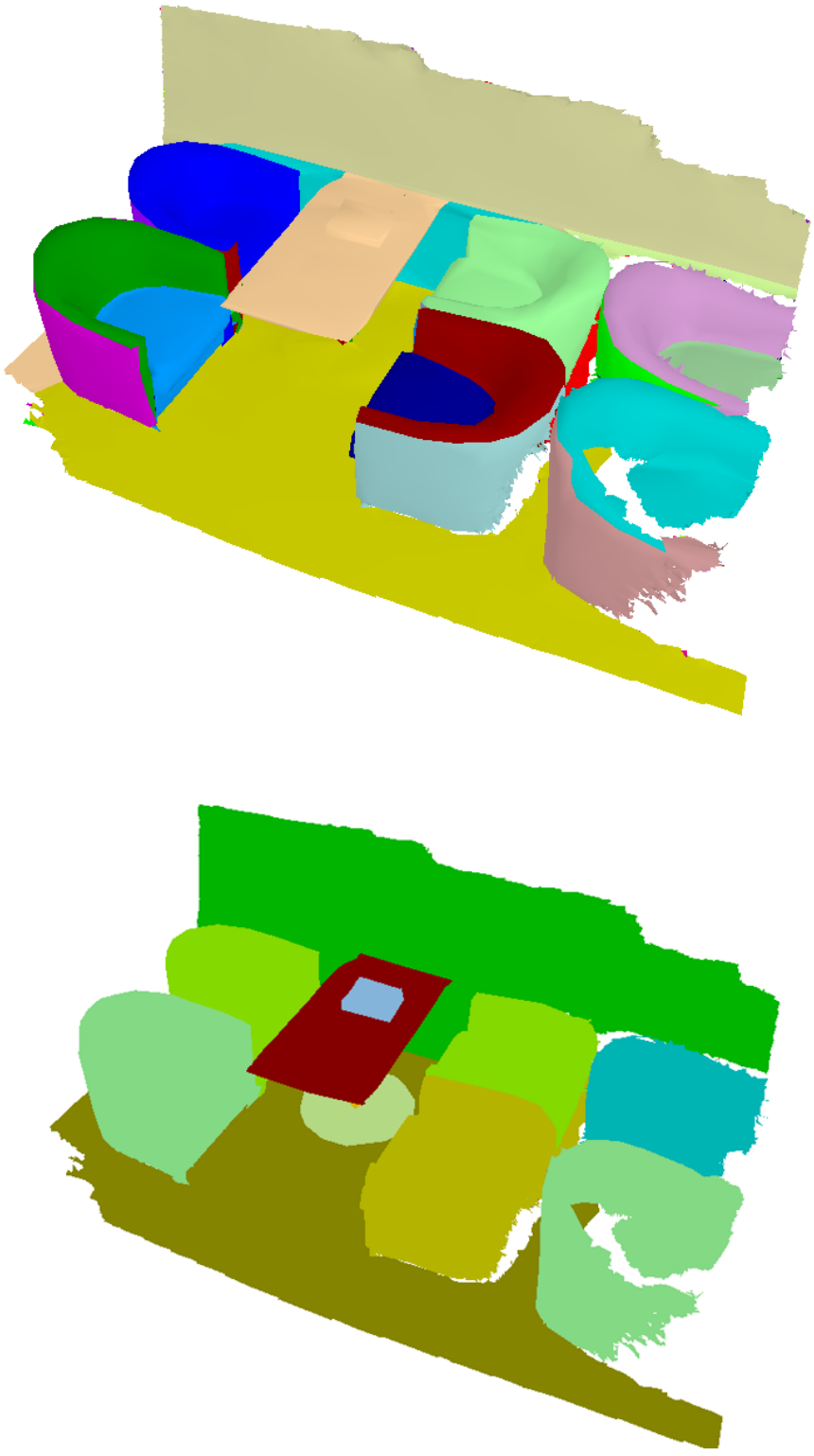

Figure 4.11: Top: Our fully-automatic segmentation results for Scene 066 with f_2 and δ_{cot}. Bottom: Our semi-automatic segmentation after user interaction.

Table 4.5: Summary of Evaluation Metrics for [1] (Over-segmented). Note that values outside the parentheses are average for 10 over-segmented results and values inside the parentheses give standard deviations

Scene ID	RI	GCE	LCE
016	0.1412 (0.0755)	0.2740 (0.1070)	0.1755 (0.0372)
057	0.1671 (0.0357)	0.3230 (0.0763)	0.1492 (0.0390)
062	0.1128 (0.1071)	0.1832 (0.0642)	0.0532 (0.0144)
066	0.0845 (0.0273)	0.2252 (0.0523)	0.1075 (0.0151)
069	0.1754 (0.0720)	0.2491 (0.0581)	0.1052 (0.0097)
071	0.1558 (0.0607)	0.2923 (0.0597)	0.1554 (0.0050)
086	0.1104 (0.0894)	0.2957 (0.0724)	0.1166 (0.0270)
109	0.0571 (0.0078)	0.3528 (0.0368)	0.1968 (0.0130)
207	0.1732 (0.1063)	0.2732 (0.0660)	0.1391 (0.0388)
223	0.1392 (0.1165)	0.2142 (0.0266)	0.1053 (0.0112)
227	0.0739 (0.0460)	0.2024 (0.0365)	0.0703 (0.0082)
234	0.1633 (0.0697)	0.2574 (0.0284)	0.1448 (0.0168)
243	0.0933 (0.0762)	0.2377 (0.0490)	0.0936 (0.0135)
249	0.2163 (0.1357)	0.1885 (0.0416)	0.0654 (0.0084)
255	0.0599 (0.0051)	0.1928 (0.0342)	0.1084 (0.0103)
273	0.0950 (0.0451)	0.3657 (0.0306)	0.1479 (0.0089)
286	0.1767 (0.1010)	0.3739 (0.0519)	0.2076 (0.0214)
522	0.3477 (0.0202)	0.2825 (0.0174)	0.1294 (0.0038)

Table 4.6: Summary of Evaluation Metrics for [1] (Final)

Scene ID	RI	GCE	LCE
016	0.2163	0.3927	0.0986
057	0.2179	0.3872	0.1391
062	0.3360	0.3607	0.1121
066	0.1257	0.2870	0.0787
069	0.2317	0.3439	0.1201
071	0.2717	0.4410	0.1841
086	0.2574	0.4015	0.1518
109	0.1163	0.4517	0.1956
207	0.2347	0.4174	0.1754
223	0.2908	0.3029	0.1420
227	0.4203	0.2218	0.0665
234	0.2883	0.3151	0.0990
243	0.0912	0.3082	0.1110
249	0.3897	0.2440	0.0771
255	0.1298	0.3147	0.1254
273	0.1460	0.4327	0.1444
286	0.2054	0.4242	0.2104
522	0.3193	0.3549	0.1363

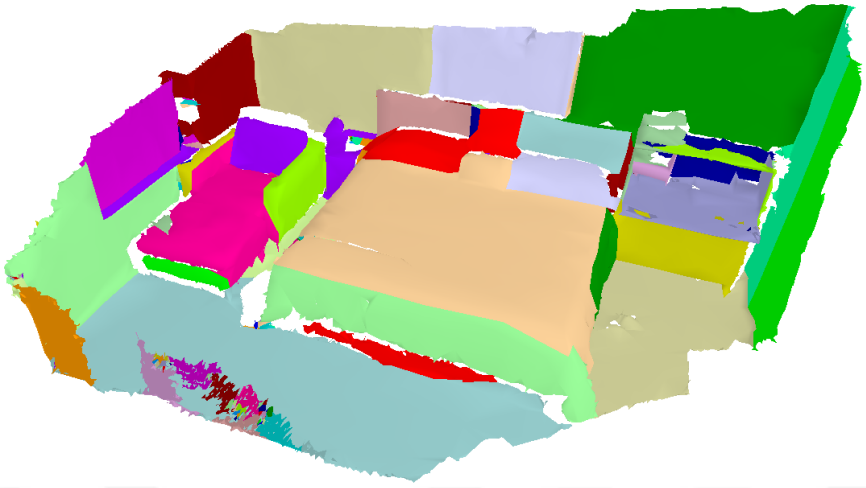

Figure 4.12: A segmentation of Scene 016 which has lower (i.e. better) GCE and LCE scores than the semi-automatic segmentation of the same scene

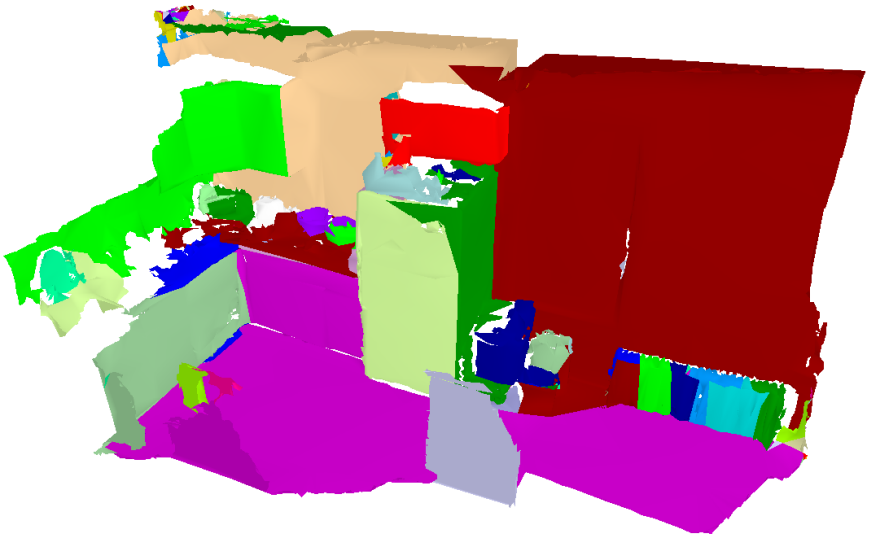

Figure 4.13: An example of a failure case. The radiator in the lower right corner is over-segmented

CHAPTER 5

CONCLUSIONS

In this thesis, we proposed a new segmentation method for indoor scenes. Our method is mainly based on applying Partial Evidence Accumulation on an ensemble created by clustering the weighted features of the scanned scene using k-medoids algorithm. We evaluated our method by comparing it with an existing approach both qualitatively and quantitatively and shared the results.

In our method, we also offer users an interactive segmentation mechanism which allows them to modify the outcome of our algorithm. Using interactive segmentation, users are able to fix improperly segmented regions without spending much time and effort. We think that makes our algorithm convenient for some practical applications. For example, creating a dataset of labeled indoor scenes is not a trivial task. Using our approach, however, one can label a large portion of scenes automatically and make necessary adjustments using interactive segmentation. Even if we do not focus on semantic segmentation, it would not be too hard to extend the application to support labeling of segments semantically.

Limitations: There are a few weak points in our method which we are aware of. First, we have no strategy to deal with the errors occur due to scanning process. For example, our method assumes that separate connected components of a scene have to be in separate segments. However, such cases may happen due to scanning quality or the geometry of the object. Another issue related with scanning errors is that if a scene contains an incomplete region (i.e. a region that is not scanned in detail), those regions become extremely noisy after the pre-processing step. The dihedral-based partitioning is also sensitive to noise. Moreover, dihedral-based partitioning has its own downsides. It depends on the initial face, for instance. In other words, if we start the dihedral-based partitioning from different faces on the scene, we may obtain different results. Thus, even if we think our intuition is appropriate, there is a lot of room for improvement. Mapping the dot products linearly, for example, may not be the ideal choice and further experiments with different mapping approaches are needed. Lastly, other kinds of features may be utilized. For example, global point signatures may be exchanged for another shape descriptor. Because it may switch the signs of the eigenvectors arbitrarily which in turn affects the distance between data points.

Future Work: Currently, we take the desired number of segments from the user. However, in literature, there are several mesh segmentation techniques which try to predict the number of segments automatically [125] [126] [60] [127]. A variation of these techniques may also be utilized for our method. Another option may be using

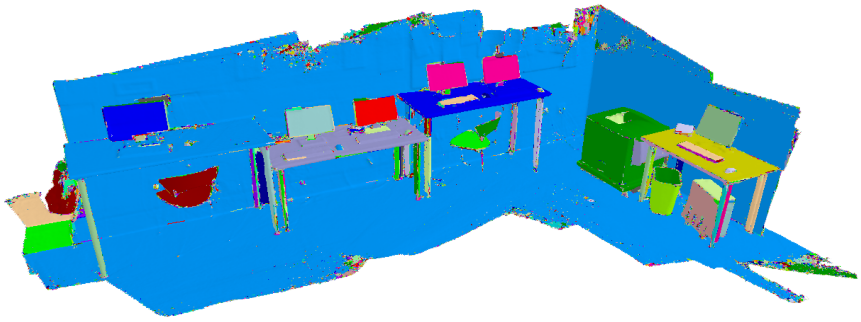

Figure 5.1: Quality of dihedral-based partitioning is apparently better on dense scenes

the dihedral-based partitioning in a different way. Currently, we run dihedral-based partitioning on the simplified version of the scene. However, it works better on raw scenes which are denser and less noisy (see Figure 5.1). Note that dihedral-based partitioning runs relatively fast even on dense scenes. As we have discussed in the limitations section, noisy regions which show up after pre-processing stage creates a number of problems. Tackling these problems would make the method more noise tolerant and improve the quality of outcomes.

In this thesis, we have tested our method with triangulated meshes. However, it is trivially extendable to other representations as long as surface normals can be calculated or predicted. In order to apply it on point clouds, for instance, tangent plane estimation [128] may be utilized.

At the interactive segmentation step, we may recommend a set of merge or split operations based on a combination of the objectness measures proposed in [1]. That would make it even faster to modify segmentations. Last but not least, a proper user study is needed for the interactive segmentation step. Moreover, there are several interactive scene segmentation tools proposed in the literature. However, to the best of our knowledge, there are no studies which compare these tools and evaluate them in a systematic way.

REFERENCES

[1] A. Karpathy, S. D. Miller, and F. Li, "Object discovery in 3d scenes via shape analysis," in *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, pp. 2088–2095, IEEE, 2013.

[2] R. Rostami, F. S. Bashiri, B. Rostami, and Z. Yu, "A survey on data-driven 3d shape descriptors," *Comput. Graph. Forum*, vol. 38, no. 1, pp. 356–393, 2019.

[3] B. Hua, Q. Pham, D. T. Nguyen, M. Tran, L. Yu, and S. Yeung, "Scenenn: A scene meshes dataset with annotations," in *Fourth International Conference on 3D Vision, 3DV 2016, Stanford, CA, USA, October 25-28, 2016*, pp. 92–101, IEEE Computer Society, 2016.

[4] A. Shamir, "A survey on mesh segmentation techniques," *Comput. Graph. Forum*, vol. 27, no. 6, pp. 1539–1556, 2008.

[5] T. Breuer, G. R. G. Macedo, R. Hartanto, N. Hochgeschwender, D. Holz, F. Hegger, Z. Jin, C. A. Mueller, J. Paulus, M. Reckhaus, J. A. Á. Ruiz, P. Plöger, and G. K. Kraetzschmar, "Johnny: An autonomous service robot for domestic environments," *J. Intell. Robotic Syst.*, vol. 66, no. 1-2, pp. 245–272, 2012.

[6] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pp. 3354–3361, IEEE Computer Society, 2012.

[7] Z. Wang, H. Liu, X. Wang, and Y. Qian, "Segment and label indoor scene based on RGB-D for the visually impaired," in *MultiMedia Modeling - 20th Anniversary International Conference, MMM 2014, Dublin, Ireland, January 6-10, 2014, Proceedings, Part I* (C. Gurrin, F. Hopfgartner, W. Hürst, H. D. Johansen, H. Lee, and N. E. O'Connor, eds.), vol. 8325 of *Lecture Notes in Computer Science*, pp. 449–460, Springer, 2014.

[8] C. Zatout, S. Larabi, I. Mendili, and S. A. E. Barnabé, "Ego-semantic labeling of scene from depth image for visually impaired and blind people," in *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019*, pp. 4376–4384, IEEE, 2019.

[9] J. P. C. Valentin, V. Vineet, M. Cheng, D. Kim, J. Shotton, P. Kohli, M. Nießner, A. Criminisi, S. Izadi, and P. H. S. Torr, "Semanticpaint: Interactive 3d labeling and learning at your fingertips," *ACM Trans. Graph.*, vol. 34, no. 5, pp. 154:1–154:17, 2015.

[10] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *2017 IEEE Conference on Computer*

Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pp. 77–85, IEEE Computer Society, 2017.

[11] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner, "Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 4578–4587, IEEE Computer Society, 2018.

[12] M. Halber, Y. Shi, K. Xu, and T. A. Funkhouser, "Rescan: Inductive instance segmentation for indoor RGBD scans," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 2541–2550, IEEE, 2019.

[13] L. Shapira, S. Shalom, A. Shamir, D. Cohen-Or, and H. Zhang, "Contextual part analogies in 3d objects," *Int. J. Comput. Vis.*, vol. 89, no. 2-3, pp. 309–326, 2010.

[14] H. Benhabiles, G. Lavoué, J. Vandeborre, and M. Daoudi, "Learning boundary edges for 3d-mesh segmentation," *Comput. Graph. Forum*, vol. 30, no. 8, pp. 2170–2182, 2011.

[15] M. Attene, S. Katz, M. Mortara, G. Patanè, M. Spagnuolo, and A. Tal, "Mesh segmentation - A comparative study," in *2006 International Conference on Shape Modeling and Applications (SMI 2006), 14-16 June 2006, Matsushima, Japan*, p. 7, IEEE Computer Society, 2006.

[16] P. Theologou, I. Pratikakis, and T. Theoharis, "A comprehensive overview of methodologies and performance evaluation frameworks in 3d mesh segmentation," *Comput. Vis. Image Underst.*, vol. 135, pp. 49–82, 2015.

[17] R. S. V. Rodrigues, J. F. M. Morgado, and A. J. P. Gomes, "Part-based mesh segmentation: A survey," *Comput. Graph. Forum*, vol. 37, no. 6, pp. 235–274, 2018.

[18] S. Izadi, R. A. Newcombe, D. Kim, O. Hilliges, D. Molyneaux, S. Hodges, P. Kohli, J. Shotton, A. J. Davison, and A. W. Fitzgibbon, "Kinectfusion: real-time dynamic 3d surface reconstruction and interaction," in *International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2011, Vancouver, BC, Canada, August 7-11, 2011, Talks Proceedings* (M. Elendt, ed.), p. 23, ACM, 2011.

[19] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *IEEE Trans. Cybern.*, vol. 43, no. 5, pp. 1318–1334, 2013.

[20] M. Billinghurst and A. Dünser, "Augmented reality in the classroom," *IEEE Computer*, vol. 45, no. 7, pp. 56–63, 2012.

[21] D. Wolf, J. Prankl, and M. Vincze, "Enhancing semantic segmentation for robotics: The power of 3-d entangled forests," *IEEE Robotics Autom. Lett.*, vol. 1, no. 1, pp. 49–56, 2016.

[22] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision, Kerkyra, Corfu, Greece, September 20-25, 1999*, pp. 1150–1157, IEEE Computer Society, 1999.

[23] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*, pp. 886–893, IEEE Computer Society, 2005.

[24] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: speeded up robust features," in *Computer Vision - ECCV 2006, 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006, Proceedings, Part I* (A. Leonardis, H. Bischof, and A. Pinz, eds.), vol. 3951 of *Lecture Notes in Computer Science*, pp. 404–417, Springer, 2006.

[25] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nat.*, vol. 521, no. 7553, pp. 436–444, 2015.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States* (P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1106–1114, 2012.

[27] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. Speech Audio Process.*, vol. 20, no. 1, pp. 30–42, 2012.

[28] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 3431–3440, IEEE Computer Society, 2015.

[29] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 146:1–146:12, 2019.

[30] A. Ioannidou, E. Chatzilari, S. Nikolopoulos, and I. Kompatsiaris, "Deep learning advances in computer vision with 3d data: A survey," *ACM Comput. Surv.*, vol. 50, no. 2, pp. 20:1–20:38, 2017.

[31] C. Nash and C. K. I. Williams, "The shape variational autoencoder: A deep generative model of part-segmented 3d objects," *Comput. Graph. Forum*, vol. 36, no. 5, pp. 1–12, 2017.

[32] M. Naseer, S. Khan, and F. Porikli, "Indoor scene understanding in 2.5/3d for autonomous agents: A survey," *IEEE Access*, vol. 7, pp. 1859–1887, 2019.

[33] H. Wang and Y. Yang, "Descent methods for elastic body simulation on the GPU," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 212:1–212:10, 2016.

[34] L. Schneider, M. Jasch, B. Fröhlich, T. Weber, U. Franke, M. Pollefeys, and M. Rätsch, "Multimodal neural networks: RGB-D for semantic segmentation and object detection," in *Image Analysis - 20th Scandinavian Conference, SCIA 2017, Tromsø, Norway, June 12-14, 2017, Proceedings, Part I* (P. Sharma and F. M. Bianchi, eds.), vol. 10269 of *Lecture Notes in Computer Science*, pp. 98–109, Springer, 2017.

[35] D. Maturana and S. A. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*, pp. 922–928, IEEE, 2015.

[36] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 1912–1920, IEEE Computer Society, 2015.

[37] L. Ma, J. Stückler, C. Kerl, and D. Cremers, "Multi-view deep learning for consistent semantic mapping with RGB-D cameras," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, pp. 598–605, IEEE, 2017.

[38] K. Guo, D. Zou, and X. Chen, "3d mesh labeling via deep convolutional neural networks," *ACM Trans. Graph.*, vol. 35, no. 1, pp. 3:1–3:12, 2015.

[39] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena, "Semantic labeling of 3d point clouds for indoor scenes," in *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain* (J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, eds.), pp. 244–252, 2011.

[40] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V* (A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), vol. 7576 of *Lecture Notes in Computer Science*, pp. 746–760, Springer, 2012.

[41] A. Martinovic, J. Knopp, H. Riemenschneider, and L. V. Gool, "3d all the way: Semantic segmentation of urban scenes from start to end in 3d," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 4456–4465, IEEE Computer Society, 2015.

[42] R. Zhang, S. A. Candra, K. Vetter, and A. Zakhor, "Sensor fusion for semantic segmentation of urban scenes," in *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pp. 1850–1857, IEEE, 2015.

[43] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena, "Labeling 3d scenes for personal assistant robots," *CoRR*, vol. abs/1106.5551, 2011.

[44] K. Xu, V. G. Kim, Q. Huang, and E. Kalogerakis, "Data-driven shape analysis and processing," *Comput. Graph. Forum*, vol. 36, no. 1, pp. 101–132, 2017.

[45] M. Meng, L. Fan, and L. Liu, "icutter: a direct cut-out tool for 3d shapes," *Comput. Anim. Virtual Worlds*, vol. 22, no. 4, pp. 335–342, 2011.

[46] Y. Zheng, C. Tai, and O. K. Au, "Dot scissor: A single-click interface for mesh segmentation," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 8, pp. 1304–1312, 2012.

[47] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, and H. Seidel, "Intelligent mesh scissoring using 3d snakes," in *12th Pacific Conference on Computer Graphics and Applications (PG 2004), 6-8 October 2004, Seoul, Korea*, pp. 279–287, IEEE Computer Society, 2004.

[48] M. Meng, L. Fan, and L. Liu, "A comparative evaluation of foreground/background sketch-based mesh segmentation algorithms," *Comput. Graph.*, vol. 35, no. 3, pp. 650–660, 2011.

[49] D. L. Page, A. F. Koschan, and M. A. Abidi, "Perception-based 3d triangle mesh segmentation using fast marching watersheds," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), 16-22 June 2003, Madison, WI, USA*, pp. 27–32, IEEE Computer Society, 2003.

[50] J. Zhang, J. Zheng, and J. Cai, "Interactive mesh cutting using constrained random walks," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 3, pp. 357–367, 2011.

[51] Y. Zheng and C. Tai, "Mesh decomposition with cross-boundary brushes," *Comput. Graph. Forum*, vol. 29, no. 2, pp. 527–535, 2010.

[52] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 169:1–169:11, 2013.

[53] D. T. Nguyen, B. Hua, L. Yu, and S. Yeung, "A robust 3d-2d interactive tool for scene segmentation and annotation," *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 12, pp. 3005–3018, 2018.

[54] P. Z. Ramirez, C. Paternesi, D. D. Gregorio, and L. di Stefano, "Shooting labels: 3d semantic labeling by virtual reality," *CoRR*, vol. abs/1910.05021, 2019.

[55] F. S. Bashiri, R. Rostami, P. L. Peissig, R. M. D'Souza, and Z. Yu, "An application of manifold learning in global shape descriptors," *Algorithms*, vol. 12, no. 8, p. 171, 2019.

[56] J. Sun, M. Ovsjanikov, and L. J. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," *Comput. Graph. Forum*, vol. 28, no. 5, pp. 1383–1392, 2009.

[57] M. Aubry, U. Schlickewei, and D. Cremers, "The wave kernel signature: A quantum mechanical approach to shape analysis," in *IEEE International Conference on Computer Vision Workshops, ICCV 2011 Workshops, Barcelona, Spain, November 6-13, 2011*, pp. 1626–1633, IEEE Computer Society, 2011.

[58] D. Raviv, M. M. Bronstein, A. M. Bronstein, and R. Kimmel, "Volumetric heat kernel signatures," in *Proceedings of the ACM workshop on 3D object retrieval, 3DOR '10, Firenze, Italy, October 25, 2010* (M. Daoudi, M. Spagnuolo, and R. C. Veltkamp, eds.), pp. 39–44, ACM, 2010.

[59] M. M. Bronstein and I. Kokkinos, "Scale-invariant heat kernel signatures for non-rigid shape recognition," in *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pp. 1704–1711, IEEE Computer Society, 2010.

[60] P. Skraba, M. Ovsjanikov, F. Chazal, and L. J. Guibas, "Persistence-based segmentation of deformable shapes," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2010, San Francisco, CA, USA, 13-18 June, 2010*, pp. 45–52, IEEE Computer Society, 2010.

[61] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. J. Guibas, "Functional maps: a flexible representation of maps between shapes," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 30:1–30:11, 2012.

[62] D. Raviv, R. Kimmel, and A. M. Bruckstein, "Graph isomorphisms and automorphisms via spectral signatures," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1985–1993, 2013.

[63] D. D. Youngster, E. Paquet, H. L. Viktor, and E. M. Petriu, "An isometry-invariant spectral approach for protein-protein docking," in *13th IEEE International Conference on BioInformatics and BioEngineering, BIBE 2013, Chania, Greece, November 10-13, 2013*, pp. 1–6, IEEE Computer Society, 2013.

[64] R. Litman and A. M. Bronstein, "Learning spectral descriptors for deformable shape correspondence," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 1, pp. 171–180, 2014.

[65] P. Heider, A. Pierre-Pierre, R. Li, and C. Grimm, "Local shape descriptors, a survey and evaluation," in *4th Eurographics Workshop on 3D Object Retrieval, 3DOR@Eurographics 2011, Llandudno, UK, April 10, 2011* (H. Laga, T. Schreck, A. Ferreira, A. Godil, I. Pratikakis, and R. C. Veltkamp, eds.), pp. 49–56, Eurographics Association, 2011.

[66] S. Tang and A. Godil, "An evaluation of local shape descriptors for 3d shape retrieval," in *Three-Dimensional Image Processing (3DIP) and Applications II, Burlingame, California, USA, January 22, 2012* (A. Baskurt and R. Sitnik, eds.), vol. 8290 of *SPIE Proceedings*, p. 82900N, SPIE, 2012.

[67] I. K. Kazmi, L. You, and J. Zhang, "A survey of 2d and 3d shape descriptors," in *10th International Conference Computer Graphics, Imaging and Visualization, CGIV 2013, Los Alamitos, CA, USA, August 6-8, 2013*, pp. 1–10, IEEE Computer Society, 2013.

[68] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–136, 1982.

[69] A. L. N. Fred and A. Lourenço, "Cluster ensemble methods: from single clusterings to combined solutions," 2008.

[70] M. C. P. de Souto, D. S. A. de Araujo, and B. L. C. da Silva, "Cluster ensemble for gene expression microarray data: Accuracy and diversity," in *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2006, part of the IEEE World Congress on Computational Intelligence, WCCI 2006, Vancouver, BC, Canada, 16-21 July 2006*, pp. 2174–2180, IEEE, 2006.

[71] D. Greene, A. Tsymbal, N. Bolshakova, and P. Cunningham, "Ensemble clustering in medical diagnostics," in *17th IEEE Symposium on Computer-Based Medical Systems (CBMS 2004), 24-25 June 2004, Bethesda, MD, USA*, pp. 576–581, IEEE Computer Society, 2004.

[72] A. P. Topchy, A. K. Jain, and W. F. Punch, "Clustering ensembles: Models of consensus and weak partitions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1866–1881, 2005.

[73] A. Strehl and J. Ghosh, "Cluster ensembles — A knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, pp. 583–617, 2002.

[74] A. L. N. Fred and A. K. Jain, "Evidence accumulation clustering based on the k-means algorithm," in *Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops SSPR 2002 and SPR 2002, Windsor, Ontario, Canada, August 6-9, 2002, Proceedings* (T. Caelli, A. Amin, R. P. W. Duin, M. S. Kamel, and D. de Ridder, eds.), vol. 2396 of *Lecture Notes in Computer Science*, pp. 442–451, Springer, 2002.

[75] A. L. N. Fred and A. K. Jain, "Data clustering using evidence accumulation," in *16th International Conference on Pattern Recognition, ICPR 2002, Quebec, Canada, August 11-15, 2002*, pp. 276–280, IEEE Computer Society, 2002.

[76] A. L. N. Fred and A. K. Jain, "Combining multiple clusterings using evidence accumulation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 835–850, 2005.

[77] A. Lourenço, S. R. Bulò, N. Rebagliati, A. L. N. Fred, M. A. T. Figueiredo, and M. Pelillo, "Consensus clustering using partial evidence accumulation," in *Pattern Recognition and Image Analysis - 6th Iberian Conference, IbPRIA 2013, Funchal, Madeira, Portugal, June 5-7, 2013. Proceedings* (J. M. Sanches, L. Micó, and J. S. Cardoso, eds.), vol. 7887 of *Lecture Notes in Computer Science*, pp. 69–78, Springer, 2013.

[78] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1997, Los Angeles, CA, USA, August 3-8, 1997* (G. S. Owen, T. Whitted, and B. Mones-Hattal, eds.), pp. 209–216, ACM, 1997.

[79] R. Huang, P. Achlioptas, L. J. Guibas, and M. Ovsjanikov, "Limit shapes - A tool for understanding shape differences and variability in 3d model collections," *Comput. Graph. Forum*, vol. 38, no. 5, pp. 187–202, 2019.

[80] M. Eisenberger, Z. Lähner, and D. Cremers, "Divergence-free shape correspondence by deformation," *Comput. Graph. Forum*, vol. 38, no. 5, pp. 1–12, 2019.

[81] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Laplacian mesh optimization," in *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia 2006, Kuala Lumpur, Malaysia, November 29 - December 2, 2006* (Y. T. Lee, S. M. H. Shamsuddin, D. Gutierrez, and N. M. Suaib, eds.), pp. 381–389, ACM, 2006.

[82] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H. Seidel, "Laplacian surface editing," in *Second Eurographics Symposium on Geometry Processing, Nice, France, July 8-10, 2004* (J. Boissonnat and P. Alliez, eds.), vol. 71 of *ACM International Conference Proceeding Series*, pp. 175–184, Eurographics Association, 2004.

[83] M. Ovsjanikov, J. Sun, and L. J. Guibas, "Global intrinsic symmetries of shapes," *Comput. Graph. Forum*, vol. 27, no. 5, pp. 1341–1348, 2008.

[84] M. Belkin, J. Sun, and Y. Wang, "Constructing laplace operator from point clouds in R^d," in *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009* (C. Mathieu, ed.), pp. 1031–1040, SIAM, 2009.

[85] O. Sorkine, "Differential representations for mesh processing," *Comput. Graph. Forum*, vol. 25, no. 4, pp. 789–807, 2006.

[86] G. Taubin, "A signal processing approach to fair surface design," in *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1995, Los Angeles, CA, USA, August 6-11, 1995* (S. G. Mair and R. Cook, eds.), pp. 351–358, ACM, 1995.

[87] U. Pinkall and K. Polthier, "Computing discrete minimal surfaces and their conjugates," *Experimental Mathematics*, vol. 2, no. 1, pp. 15–36, 1993.

[88] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, "Implicit fairing of irregular meshes using diffusion and curvature flow," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1999, Los Angeles, CA, USA, August 8-13, 1999* (W. N. Waggenspack, ed.), pp. 317–324, ACM, 1999.

[89] G. Taubin, "Geometric signal processing on polygonal meshes," in *21st Annual Conference of the European Association for Computer Graphics, Eurographics 2000 - State of the Art Reports, Interlaken, Switzerland, August 21-25, 2000*, Eurographics Association, 2000.

[90] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, "Discrete differential-geometry operators for triangulated 2-manifolds," in *Third International Workshop "Visualization and Mathematics", VisMath 2002, Berlin, Germany, May*

22-25, 2002 (H. Hege and K. Polthier, eds.), Mathematics and Visualization, pp. 35–57, Springer, 2002.

[91] G. Xu, "Discrete laplace-beltrami operators and their convergence," *Comput. Aided Geom. Des.*, vol. 21, pp. 767–784, 2004.

[92] Z. Karni and C. Gotsman, "Spectral compression of mesh geometry," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2000, New Orleans, LA, USA, July 23-28, 2000* (J. R. Brown and K. Akeley, eds.), pp. 279–286, ACM, 2000.

[93] G. Xu, "Discrete laplace-beltrami operator on sphere and optimal spherical triangulations," *Int. J. Comput. Geom. Appl.*, vol. 16, no. 1, pp. 75–93, 2006.

[94] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, "Discrete differential-geometry operators in n d," 2000.

[95] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[96] B. Lévy, "Laplace-beltrami eigenfunctions towards an algorithm that "understands" geometry," in *2006 International Conference on Shape Modeling and Applications (SMI 2006), 14-16 June 2006, Matsushima, Japan*, p. 13, IEEE Computer Society, 2006.

[97] H. Zhang, O. van Kaick, and R. Dyer, "Spectral mesh processing," *Comput. Graph. Forum*, vol. 29, no. 6, pp. 1865–1894, 2010.

[98] R. M. Rustamov, "Laplace-beltrami eigenfunctions for deformation invariant shape representation," in *Proceedings of the Fifth Eurographics Symposium on Geometry Processing, Barcelona, Spain, July 4-6, 2007* (A. G. Belyaev and M. Garland, eds.), vol. 257 of *ACM International Conference Proceeding Series*, pp. 225–233, Eurographics Association, 2007.

[99] V. Jain and H. Zhang, "Robust 3d shape correspondence in the spectral domain," in *2006 International Conference on Shape Modeling and Applications (SMI 2006), 14-16 June 2006, Matsushima, Japan*, p. 19, IEEE Computer Society, 2006.

[100] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007* (N. Bansal, K. Pruhs, and C. Stein, eds.), pp. 1027–1035, SIAM, 2007.

[101] L. Kaufman and P. J. Rousseeuw, "Clustering by means of medoids," 1987.

[102] H. Park and C. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3336–3341, 2009.

[103] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[104] E. Rodolà, S. R. Bulò, and D. Cremers, "Robust region detection via consensus segmentation of deformable shapes," *Comput. Graph. Forum*, vol. 33, no. 5, pp. 97–106, 2014.

[105] A. Lourenço, S. R. Bulò, N. Rebagliati, A. L. N. Fred, M. A. T. Figueiredo, and M. Pelillo, "Probabilistic consensus clustering using evidence accumulation," *Mach. Learn.*, vol. 98, no. 1-2, pp. 331–357, 2015.

[106] G. Guennebaud and B. Jacob, "Eigen v3." http://eigen.tuxfamily.org, 2010.

[107] R. B. Lehoucq, D. C. Sorensen, and C. Yang, "Arpack users guide: Solution of large scale eigenvalue problems by implicitly restarted arnoldi methods.," 1997.

[108] Y. Qiu, "Spectra (sparse eigenvalue computation toolkit as a redesigned arpack)," 2015.

[109] N. Silberman and R. Fergus, "Indoor scene segmentation using a structured light sensor," in *IEEE International Conference on Computer Vision Workshops, ICCV 2011 Workshops, Barcelona, Spain, November 6-13, 2011*, pp. 601–608, IEEE Computer Society, 2011.

[110] S. Song, S. P. Lichtenberg, and J. Xiao, "SUN RGB-D: A RGB-D scene understanding benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 567–576, IEEE Computer Society, 2015.

[111] J. Xiao, A. Owens, and A. Torralba, "SUN3D: A database of big spaces reconstructed using sfm and object labels," in *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pp. 1625–1632, IEEE Computer Society, 2013.

[112] K. Lai, L. Bo, and D. Fox, "Unsupervised feature learning for 3d scene labeling," in *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, pp. 3050–3057, IEEE, 2014.

[113] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pp. 573–580, IEEE, 2012.

[114] P. Hanrahan, "Scenegrok: inferring action maps in 3d environments," *ACM Trans. Graph.*, vol. 33, no. 6, pp. 212:1–212:10, 2014.

[115] A. Handa, V. Patraucean, V. Badrinarayanan, S. Stent, and R. Cipolla, "Scenenet: Understanding real world indoor scenes with synthetic data," *CoRR*, vol. abs/1511.07041, 2015.

[116] M. Firman, "RGBD datasets: Past, present and future," in *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2016, Las Vegas, NV, USA, June 26 - July 1, 2016*, pp. 661–673, IEEE Computer Society, 2016.

[117] G. Gerig, M. Jomier, and M. Chakos, "Valmet: A new validation tool for assessing and improving 3d object segmentation," in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2001, 4th International Conference, Utrecht, The Netherlands, October 14-17, 2001, Proceedings* (W. J. Niessen and M. A. Viergever, eds.), vol. 2208 of *Lecture Notes in Computer Science*, pp. 516–523, Springer, 2001.

[118] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe, "Texture mapping progressive meshes," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001, Los Angeles, California, USA, August 12-17, 2001* (L. Pocock, ed.), pp. 409–416, ACM, 2001.

[119] H. Benhabiles, J. Vandeborre, G. Lavoué, and M. Daoudi, "A framework for the objective evaluation of segmentation algorithms using a ground-truth of human segmented 3d-models," in *IEEE International Conference on Shape Modeling and Applications, SMI 2009, Beijing, China, 26-28 June 2009* (J. Yong, M. Spagnuolo, and W. Wang, eds.), pp. 36–43, IEEE Computer Society, 2009.

[120] X. Chen, A. Golovinskiy, and T. A. Funkhouser, "A benchmark for 3d mesh segmentation," *ACM Trans. Graph.*, vol. 28, no. 3, p. 73, 2009.

[121] Q. Huang and B. Dom, "Quantitative methods of evaluating image segmentation," in *Proceedings 1995 International Conference on Image Processing, Washington, DC, USA, October 23-26, 1995*, pp. 53–56, IEEE Computer Society, 1995.

[122] D. R. Martin, C. C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings of the Eighth International Conference On Computer Vision (ICCV-01), Vancouver, British Columbia, Canada, July 7-14, 2001 - Volume 2*, pp. 416–425, IEEE Computer Society, 2001.

[123] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971.

[124] B. Peng, L. Zhang, and D. Zhang, "A survey of graph theoretical approaches to image segmentation," *Pattern Recognit.*, vol. 46, no. 3, pp. 1020–1038, 2013.

[125] S. Katz, G. Leifman, and A. Tal, "Mesh segmentation using feature point and core extraction," *Vis. Comput.*, vol. 21, no. 8-10, pp. 649–658, 2005.

[126] L. Shapira, A. Shamir, and D. Cohen-Or, "Consistent mesh partitioning and skeletonisation using the shape diameter function," *Vis. Comput.*, vol. 24, no. 4, pp. 249–259, 2008.

[127] R. Litman, A. M. Bronstein, and M. M. Bronstein, "Diffusion-geometric maximally stable component detection in deformable shapes," *Comput. Graph.*, vol. 35, no. 3, pp. 549–560, 2011.

[128] H. Hoppe, T. DeRose, T. Duchamp, J. A. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *Proceedings of the 19th Annual*

Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1992, Chicago, IL, USA, July 27-31, 1992 (J. J. Thomas, ed.), pp. 71–78, ACM, 1992.